



МИНИСТЕРСТВО ПРОСВЕЩЕНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ЮЖНО-УРАЛЬСКИЙ ГОСУДАРСТВЕННЫЙ
ГУМАНИТАРНО-ПЕДАГОГИЧЕСКИЙ УНИВЕРСИТЕТ»
(ФГБОУ ВО «ЮУрГГПУ»)

ФИЗИКО-МАТЕМАТИЧЕСКИЙ ФАКУЛЬТЕТ
КАФЕДРА ИНФОРМАТИКИ, ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И МЕТОДИКИ
ОБУЧЕНИЯ ИНФОРМАТИКИ

Методические особенности изучения современных языков
программирования в курсе информатики средней школы
Выпускная квалификационная работа по направлению
44.03.01 Педагогическое образование

Направленность программы бакалавриата
«Информатика»
Форма обучения заочная

Проверка на объем заимствований:

61,97 % авторского текста

Работа рекомендована к защите

26 июня 2021 г.

Зав. кафедрой ИИТиМОИ

А. А. Рузакова

Выполнил:

Студент группы ЗФ 513-092-5-1 ДВ

Смирнов Александр Сергеевич

Научный руководитель:

к. п. н., доцент кафедры ИИТиМОИ

Носова Людмила Сергеевна

Челябинск
2021



МИНИСТЕРСТВО ПРОСВЕЩЕНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ЮЖНО-УРАЛЬСКИЙ ГОСУДАРСТВЕННЫЙ
ГУМАНИТАРНО-ПЕДАГОГИЧЕСКИЙ УНИВЕРСИТЕТ»
(ФГБОУ ВО «ЮУрГГПУ»)

ФИЗИКО-МАТЕМАТИЧЕСКИЙ ФАКУЛЬТЕТ
КАФЕДРА ИНФОРМАТИКИ, ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И МЕТОДИКИ
ОБУЧЕНИЯ ИНФОРМАТИКИ

**Методические особенности изучения современных языков
программирования в курсе информатики средней школы**
Выпускная квалификационная работа по направлению
44.03.01 Педагогическое образование

Направленность программы бакалавриата
«Информатика»
Форма обучения заочная

Проверка на объем заимствований:
_____ % авторского текста
Работа рекомендована к защите
_____ 2021 г.
Зав. кафедрой ИИТиМОИ
А. А. Рузакова

Выполнил:
Студент группы ЗФ 513-092-5-1
Смирнов Александр Сергеевич
Научный руководитель:
к. п. н., доцент кафедры ИИТиМОИ
Носова Людмила Сергеевна

Челябинск
2021

ОГЛАВЛЕНИЕ

Введение.....	3
Глава 1. Теоретический анализ методических особенностей изучения современных языков программирования в курсе информатики средней школы	6
1.1 Понятие «язык программирования»	6
1.2 Современные языки программирования и требования к выбору языка	13
1.3 Особенности обучению программирования в курсе информатики средней школы.....	19
Вывод по главе 1	27
Глава 2. Методические особенности изучения языков программирования в школе.....	28
2.1 Элективный курс «Основы программирования на python»	28
2.2 Методическое содержание элективного курса «Основы программирования на Python».....	32
2.3 Апробация результатов исследования в школе	58
Выводы по главе 2:	60
Заключение	61
Список использованных источников	63
Приложение	69

ВВЕДЕНИЕ

Актуальность исследования. В современном обществе наблюдается тенденция быстрого развития информационных технологий. Уследить за всеми изменениями в данной области очень сложно, но необходимо для дальнейшего ее и своего развития. Переход к постиндустриальному обществу требует от выпускников знать не только основы алгоритмизации и программирования, но также применять свои знания на практике, используя несколько языков. Но большинство языков высокого уровня сложны в освоении ввиду особенностей синтаксиса.

Согласно Федерального государственного образовательного стандарта основного общего образования (далее: ФГОС ООО) изучение предметной области «Математика и информатика» должно обеспечить сформированность основ логического, алгоритмического и математического мышления, а также сформированность умений применять полученные знания при решении различных задач.

В требованиях к предметным результатам освоения базового и профильного курса информатики также представлена область алгоритмизации и программирования.

Основными, наиболее популярными современными языками программирования, которые изучаются в школьном курсе информатики являются: R, Python, C#, C++ и Java. С. Отметим, что на законодательном уровне выбор изучаемого в школе языка программирования не регламентируется. Однако существуют параметры, которым изучаемый язык должен соответствовать для достижения образовательных целей, установленных федеральным стандартом.

Python наиболее приближен к алгоритмическому языку программирования. Не имеет строгих синтаксических правил, является более современным языком программирования, чем вызывает больший интерес у учащихся.

Несмотря на важность и актуальность изучения современных языков программирования в условиях реализации Федерального государственного образовательного стандарта в учебном плане основного образования на освоение базового курса информатики и ИКТ предлагается только один час в неделю. Вследствие этого, возникают большие проблемы в процессе овладения темы «Программирования» на уроках информатики, это происходит по причине объемного содержания темы и небольшого количества часов на её изучения.

Актуальность, социальная значимость и недостаточно методическая разработанность данной проблемы определила выбор темы исследования: «Методические особенности изучения современных языков программирования в курсе информатики средней школы».

Цель работы: изучить теоретические аспекты темы исследования и разработать элективный курс «Основы программирования на Python».

Объект исследования: процесс изучения современных языков программирования в курсе информатики средней школы.

Предмет исследования: методические особенности процесса изучения языка программирования Python в средней школе.

Задачи исследования:

1. Изучить понятие «язык программирования» в психолого-педагогической и методической литературе.
2. Проанализировать современные языки программирования и требования к выбору языка.
3. Рассмотреть особенности обучению программирования в курсе информатики средней школы.
4. Разработать методическое содержание элективного курса «Основы программирования на Python».
5. Провести апробацию элективного курса.

Экспериментальная база исследования: Муниципальное казенное общеобразовательное учреждение «Основная общеобразовательная школа № 19», р.п. Роза.

Для реализации поставленных задач использовались следующие **методы** исследования:

1. Теоретические: изучение и анализ психолого-педагогической литературы по проблеме исследования, анализ, сравнение и обобщение результатов работы.

2. Практические: разработка продукта, педагогическое наблюдение и рефлексия.

Гипотеза: если в программу среднего общего образования, разработать и включить курс «Основы программирования на Python», то это будет способствовать овладению учащимися основами логического и алгоритмического мышления, формировать базовые понятия структурного программирования, развивать аналитический стиль мышления начинающих программистов, а также умение самостоятельно составлять алгоритмы решения задач и формировать правильный стиль мышления.

Практическая значимость исследования заключается в том, что разработанные нами материалы могут быть использованы учителями информатики в средней школе на уроках информатики или в рамках внеурочной деятельности.

Структура работы состоит из введения, двух глав с выводами, заключения, списка использованной литературы и приложения. Текст работы иллюстрирован таблицами и рисунками, отражающими основные положения и результаты.

ГЛАВА 1. ТЕОРЕТИЧЕСКИЙ АНАЛИЗ МЕТОДИЧЕСКИХ ОСОБЕННОСТЕЙ ИЗУЧЕНИЯ СОВРЕМЕННЫХ ЯЗЫКОВ ПРОГРАММИРОВАНИЯ В КУРСЕ ИНФОРМАТИКИ СРЕДНЕЙ ШКОЛЫ

1.1 Понятие «язык программирования»

Любая программа, основанная на алгоритме, представляет собой набор последовательно построенных команд, которые должны быть определены и реализованы исполнителем (машиной) для достижения поставленной цели. Как утверждал А. Н. Бобров, трудность заключается в том, что машина, не может понять естественный язык, поэтому были разработаны языки, называемые языками программирования, которые используются для того, чтобы «учить» компьютер. Свойства языков программирования однозначны при формировании слов (нельзя менять слова в выражении), а также отличаются ограниченным набором командных слов. Другими словами, языки программирования являются формальными [3, с. 61].

Программирование (Programming language) – это раздел информатики, призванный изучать вопросы разработки программного обеспечения современных электронно-вычислительных машин. В наиболее узком смысле, программирование – это процесс создания алгоритмических действий языках программирования [45, с. 97].

В. В. Малеев считает, что формирование практических компьютерных проектов общепринято именовать практическим программированием [22].

Изучение программирования в базовом курсе информатики общеобразовательной школы, с точки зрения М. П. Лапчик, И. Г. Семакин, Е. К. Хенер, имеет два аспекта:

– развивающий аспект: имеется в виду формирование алгоритмического мышления школьников;

– программистский аспект: понимают развитие навыков составления учебных компьютерных программ [17].

В процессе реализации первого аспекта школьникам даются наиболее общие представления о том, что такое языки программирования, что представляет собой написание программы на языках программирования, как создается программа для различных областей знаний. Также этот аспект связан с усилением основательного элемента базисного направления информатики. Следующий подход ориентирован напрямую в высокочувствительную подготовку подростков. Исследование программирования, бесспорно, является помощью подросткам дать оценку собственные возможности к специальности разработчика программного обеспечения, таким образом, равно как данная специальность достаточно все распространена и престижна в наше период.

Вслед за Н. Д. Угриновичем, под языком программирования будем понимать «формализованный язык для описания алгоритма решения задачи на компьютере» [42].

Р. В. Наумов в своей статье приводит следующую краткую историю программирования: языки программирования начинают свою историю с первой в мире программистки Ады Лавлейс (Августа Ада Кинг, графиня Лавлейс, математик). Ада Лавлейс родилась 10 декабря 1815 г. в Лондоне. Она принимала участие в разработке первой вычислительной машины - механическая машина Ч. Бэббиджа и является создателем первой программы для нее. Ада Лавлейс впервые ввела в употребление термины «цикл» и «рабочая ячейка». В честь нее в 1975 году был назван язык программирования Ада. Бурное развитие языков программирования приходится на 1945-1955 гг., когда появились первые ЭВМ (Электронные Вычислительные Машины), для которых программы составлялись вначале на машинном языке, а затем и на Ассемблере – мнемоническом представлении машинного языка [23].

Е. Ф. Родыгин утверждает, что язык высокого уровня (или

высокоуровневый язык) – это язык программирования, наиболее приближенный к человеческому языку. Он содержит смысловые конструкции, описывает структуры данных, выполняет над ними различные операции [34].

Язык высокого уровня (согласно ГОСТ 19781-90) язык программирования, понятия и структура которого удобны для восприятия человеком. Языки высокого уровня отражают потребности программиста, но не возможности системы обработки данных [6].

В 1954 году появился первый язык программирования высокого уровня Фортран, и началась новая эра развития программирования.

Вскоре после языка Fortran появились такие ныне широко известные языки, как Algol, Cobol, Basic, PL/1, Pascal, APL, ADA, C, Forth, Lisp, Modula и др. В настоящее время насчитывается свыше 2000 различных языков высокого уровня [34].

В развитии инструментального программного обеспечения (т.е. программного обеспечения, служащего для создания программных средств в любой проблемной области) рассматривают пять поколений языков программирования (табл. 1).

Таблица 1 – Эволюция языков программирования

Поколение	Язык программирования	Характеристика
Первое	Машинные	Ориентированы на использование в конкретной ЭВМ, сложны в освоении, требуют хорошего знания архитектуры ЭВМ
Второе	Ассемблеры, Макроассемблеры	Более удобны для использования, но по-прежнему машинно-зависимы
Третье	Языки высокого уровня	Мобильные, человеко-ориентированные, проще в освоении
Четвертое	Непроцедурные, объектно-ориентированные, языки запросов, параллельные	Ориентированы на непрофессионального пользователя и на ЭВМ с параллельной архитектурой
Пятое	Языки искусственного интеллекта, экспертных систем и баз знаний, естественные языки	Ориентированы на повышение интеллектуального уровня ЭВМ и интерфейса с языками

Ассемблеры все еще используются сегодня, потому что программная

система (аппаратный сервис), написанная на них, работают быстрее, чем аналогичные программы, написанные на прочих языках программирования.

После ассемблеров наступил рассвет так называемых высокоуровневых языков. Для этих языков необходимо было разработать более сложные переводчики. Программные коды, написанные на языках высокого уровня, имеют логическую структуру, что позволяет облегчить разработку и корректировку программы. В отличие от областей, которые все еще ограничены типом машины, это разновидности языков высокого уровня могут быть адаптированы под любого исполнителя. Другими словами, написав программу один раз, программист может реализовать ее на любом компьютере [1, с. 16].

Машинный язык – это единственный способ общаться с компьютерами. Первые написанные программы были именно такими, потому что не было другого способа «контактировать» между человеком и компьютером. Каждая команда машинного языка выполняется определенным электронным устройством. Данные и команды записываются в цифровом виде (например, в шестнадцатеричной или двоичной кодировке). Человеку очень трудно понять программу на этом языке, и даже небольшая программа будет состоять из многих строк кода. Также, в интерпретации машинного кода нет единства, так как каждая машина могла понимать различные комбинации нулей и единиц. Программа, написанная для одного компьютера, могла не работать на другом [34].

Следующим важным шагом в развитии программирования является появление объектно-ориентированных языков. Они отличаются от языков высокого уровня тем, что их можно отделить от алгоритма реализации программы. Разработчик использует эти языки для работы с виртуальными объектами. Сегодня крупные и сложные проекты реализуются в первую очередь используя объектно-ориентированные языки программирования, такие как C#, C++, Perl, PHP [14, с. 358].

Современные языки высокого уровня оперируют уже целыми

объектами – сложными конструкциями, обладающими определенным состоянием и поведением.

К интенсивно развивающемуся в настоящее время пятому поколению относятся языки искусственного интеллекта, экспертных систем, баз знаний (InterLisp, ExpertLisp, IQLisp, SAIL и др.), а также естественные языки, не требующие освоения какого-либо специального синтаксиса (в настоящее время успешно используются естественные языки программирования с ограниченными возможностями – Clout, Q&A, HAL и др.) [34].

Далее перейдем к классификации языков программирования. Сразу отметим, что, как правило, каких-либо определенных факторов классификации в литературе не выявлено. Но на основании предложенных классификаций можно составить сводную таблицу, с обозначением наиболее часто встречаемых факторов, с приведением примеров языков программирования (табл. 2).

Таблица 2 – Классификация языков программирования

Фактор	Характеристика	Группы	Примеры языка программирования
Уровень языка программирования	Степень близости языка программирования к архитектуре компьютера	Низкий	Автокод, ассемблер
		Высокий	Fortran, Pascal, ADA, Basic, Сидр. ЯВУ
		Сверхвысокий	Сетл
Специализация языка программирования	Потенциальная или реальная область применения	Общего назначения	Algol, PL/1, Simula, Basic, Pascal
		Специализированные	Fortran (инженерные расчеты), Cobol (коммерческие задачи), Refal, Lisp (символьная обработка), Modula, Ada (программирование в реальном времени)
Алгоритмичность (процедурность)	Возможность абстрагироваться от деталей алгоритма решения задачи. Алгоритмичность тем выше, чем точнее приходится планировать порядок выполняемых действий	процедурные	Ассемблер, Fortran, Basic, Pascal, Ada
		непроцедурные	Prolog, Langin

При конструировании алгоритмического языка необходимо: исходить

из некоторого набора вычислительных структур (структур данных); вводить как составные операции, так и составные (структурированные) объекты; выбрать форму, удобную как для человека, который описывает алгоритм, так и для человека, который должен будет читать и понимать этот алгоритм, – форму, соответствующую кругу человеческих понятий и представлений. М. Н. Бородин выделяет три составляющие (любого) языка: алфавит, синтаксис и семантика.

Алфавит – набор основных символов, «букв алфавита», никакие другие символы в предложениях языка не допускаются.

Синтаксис – система правил, определяющая допустимые конструкции из букв алфавита. Синтаксис отвечает на вопрос: является ли последовательность символов текстом (программой) на данном языке или нет?

Семантика – система правил, истолковывающая отдельные конструкции языка с точки зрения процесса обработки (смысл программы) [4].

Как уже было описано выше, при описании языка и при построении алгоритма используются понятия языка. Любое понятие алгоритмического языка имеет свой синтаксис и свою семантику. Синтаксические правила показывают, как образуется данное понятие из других понятий и (или) букв алфавита. Семантические правила определяют свойства данного понятия в зависимости от свойств, используемых в них понятий.

Язык программирования задается своим описанием и реализуется в виде специальной программы: компилятора или интерпретатора.

Рассмотрим основные понятия языка программирования.

Декларативный язык программирования (от лат. *declaratio* – объявление) – язык программирования высокого уровня, построенный: на описании данных; и на описании искомого результата. Декларативные языки подразделяются на функциональные и логические языки [19].

Исходный код – текст программы на алгоритмическом языке. В

компьютере исходный текст либо непосредственно выполняется интерпретатором, либо предварительно переводится компилятором в стандартный загрузочный код, способный многократно исполняться в определенной вычислительной среде [20].

Машинный язык – язык программирования, элементами которого являются команды компьютера, характеризующиеся: количеством операндов в команде; назначением информации, задаваемой в операндах; набором операций, которые может выполнить компьютер и др. Конструкции машинного языка интерпретируются непосредственно аппаратурой [34].

Переменная – в языках программирования – именованная часть памяти, в которую могут помещаться разные значения переменной. Причем в каждый момент времени переменная имеет единственное значение. В процессе выполнения программы значение переменной может изменяться. Тип переменных определяется типом данных, которые они представляют.

Процедурно-ориентированный язык программирования – язык программирования высокого уровня, в основу которого положен принцип описания (последовательности) действий, позволяющей решить поставленную задачу. Обычно процедурно-ориентированные языки задают программы как совокупности процедур или подпрограмм. Примеры: BASIC, PASCAL, FORTRAN, Си [5].

Более подробно языки программирования будут рассмотрены в последующих пунктах.

Таким образом, программирование – процесс создания компьютерных программ. В узком смысле под программированием понимается написание инструкций (программ) на конкретном языке программирования (часто по уже имеющемуся алгоритму- плану, методу решения поставленной задачи).

1.2 Современные языки программирования и требования к выбору языка

Рассмотрим и сравним пять самых распространенных и востребованных языков программирования.

Язык программирования Visual Basic, имеет такие достоинства, как простой синтаксис (позволяющий очень быстро освоить язык), возможность использования методов визуального проектирования и объектно-ориентированного подхода, а также широкий диапазон приложений, работающих в сети Интернет.

Недостатками языка Visual Basic являются низкая скорость работы написанных программ, отсутствие механизма наследования реализации объектов, не рационально высокая затрата времени для разработки сложных программ, а при большом количестве данных, может не хватить физической памяти [18, с. 232].

Java известен своей мультиплатформенностью и тем, что находится в составе большинства современных операционных систем, так как работа многих приложений без него будет недостаточно результативной или вообще невозможной. Разработка Java началась в 1990 году, первая официальная версия – Java 1.0, – была выпущена только 21 января 1996 года. Практически каждый пользователь сталкивался с необходимостью установить или обновить Java-модуль. К подводным камням Java относятся медлительность написанных на нём программ и их «прожорливость» (то есть они задействуют оперативную память в большом объёме). Системами программирования являются: Idea, Eclipse, NetBeans, JDeveloper, Android Studio.

Образовательными возможностями являются: создание приложений, игр и веб-сайтов, системные приложения, приложения для мобильных телефонов [44, с. 892].

Далее рассмотрим программирование C++, который был разработан в 1983 г. Автор Страуструп, Бьёрн. Работая с языком программирования C++

можно выделить следующий ряд достоинств: Возможность работы на низком уровне с памятью, адресами, портами. У программиста есть максимальный контроль над всеми аспектами структуры и порядка исполнения программы., можно выделить эффективность, так же масштабируемость, которая позволяет разрабатывать на языке C++ программы для самых различных платформ и систем. Возможность создания обобщенных алгоритмов для разных типов данных, их специализация, и вычисления на этапе компиляции, используя шаблоны. Доступность компилирующих программ. Возможность создания обобщённых контейнеров и алгоритмов для разных типов данных, их специализация и вычисления на этапе компиляции, используя шаблоны. Кроссплатформенность. Доступны компиляторы для большого количества платформ, на языке C++ разрабатывают программы для самых различных платформ и систем. Поддержка новых парадигм программирования. Предсказуемость выполнения программ, что является важным для систем реального времени. Автоматический вызов деструктора (Код который выполняется при завершении программы) Управления константами. Построение шаблонов. Возможность множественного наследия интерфейса и реализации. Высокая совместимость с языком программирования Си, а также вычислительная мощь от языка программирования Си, что дает возможность существующий код на Си с минимальными изменениями использовать на C++. При помощи шаблонов и множественного наследия можно имитировать комбинаторную параметризацию библиотек. Возможность расширения языка для поддержки парадигм. Возможность встраивать предметно-ориентированных языков программирования в сам код. Возможность создавать пользовательские функции - операторы, позволяющие записывать выражения над пользовательскими типами в алгебраической форме [34].

Конечно, не смотря на такое количество положительных черт, у языка C++ имеются свои недостатки, например плохо продуманный синтаксис,

который схож с языком, так же примитивный, унаследованный от Си процессор. Не интуитивные преобразования некоторых типов, недостаточность информации о типах данных, во время компиляции. А использование шаблонов может привести к коду большого объема.

Системами программирования являются: Visualstudio, Eclipse CDT, NetBeans, CodeLite, Code::Blocks.

Образовательными возможностями являются: создание приложений, игр, математические расчеты, нейросети.

Рассмотрим программирование C#. Системами программирования являются: Visualstudio, Project Rider, Eclipse, MonoDevelop, Code::Blocks. Образовательными возможностями являются: создание приложений, игр и веб-сайтов [34].

Изучая язык программирования Pascal, хотелось бы отметить следующие достоинства: чрезвычайно легкий синтаксис и небольшое число базовых понятий, что делает работу с ним крайне несложной. Невысокие аппаратные и системные требования компилятора, что можно сказать и о софте, который будет выполнен с помощью этого языка программирования. Создание за кратчайшее время оконную утилиту, а кроме того, любая утилита, разработанная на этом языке, будет легко читаться. Открытый исходный код, который доступен к редактированию [44, с. 892].

Не смотря на такое количество достоинств, у Паскаля имеется следующий ряд недостатков: Pascal считается устаревшим языком, так как создан в 60-х годах, помимо этого компилятор рассчитан на реальный режим dos, который сейчас практически не используется, что является одной из причин очень малого количества действительного программного обеспечения.

Хотя язык Pascal является самым распространенным учебным структурным языком программирования, существуют и другие языки, не чем не уступающие ему, но также имеющие еще более простой синтаксис, например, язык Python.

Python наиболее приближен к алгоритмическому языку программирования. Первая стабильная версия появилась в январе 1994 года. Не имеет строгих синтаксических правил, является более современным языком программирования, чем вызывает больший интерес у учащихся [39, с. 15].

Python – весьма популярный в наше время язык, созданный с целью как можно более простого написания сложных программ. Он был образован на основе ранних разработок и впитал в себя все их достижения. При этом постоянно выходят новые обновления, с каждым из которых он становится всё совершеннее. К основным плюсам Python относятся простота и многофункциональность. Однако за многофункциональностью скрывается низкая скорость исполнения, а за простотой – порой невнятный системный код, зачастую содержащий множество ошибок [31].

В 2016 году была создана версия MicroPython для BBC Micro Bit как часть вклада Python Software Foundation в партнерство Micro Bit с BBC. В июле 2017 года MicroPython был разветвлен для создания CircuitPython, версии MicroPython с акцентом на образование и простоту использования. MicroPython и CircuitPython поддерживают несколько разные наборы аппаратного обеспечения (например, CircuitPython поддерживает платы Atmel SAM D21 и D51, но не поддерживает ESP8266). Начиная с версии 4.0, CircuitPython основан на версии 1.9.4 MicroPython. В 2017 году Microsemi создала порт MicroPython для архитектуры RISC-V (RV32 и RV64). В апреле 2019 года была создана версия MicroPython для Lego Mindstorms EV3.

Системами программирования являются: Python ide, PyCharm, Visual Studio, Spyder, Thonny [33].

Образовательными возможностями являются: создание приложений, игр и веб-сайтов, математические расчеты, нейросети, системные приложения, операционные системы.

Проведем анализ современных языков программирования (табл. 3).

Таблица 3 – Анализ современных языков программирования

Язык программирования	Python	C++	C#	Java	Pascal
Сходство структуры алгоритмическим языком	+	-	-	-	+
Поддержка структурной парадигмы программирования	+	+	+	+	+
Поддержка объектно-ориентированной парадигмы программирования	+	+	+	+	В некоторых версиях языка. В частности, версия языка PascalABC.Net поддерживает объектно-ориентированную парадигму программирования
Читаемость и понятность кода	+	-	-	+	+
Типизация данных	При создании переменной ее тип не указывается в явном виде, но позже может быть преобразован к какому-либо типу специальными функциями.	Типы данных указываются при создании переменных, но переменные могут быть преобразованы к другому типу.	При создании переменной необходимо в явном виде указать ее тип. Тип переменной нельзя изменять в ходе работы программы.		
Возможность работы с массивами	Массивов как таковых не существует, однако есть такой тип данных, как списки. В определенной мере, списки подобны массивам в С. Единственной разницей является то, что элементы одного списка могут иметь разные типы данных	+	+	+	+

Несмотря на единые требования образовательного стандарта, у каждого учителя свой набор требований к языку программирования. Это может быть удобная, дружелюбная среда разработки; простой,

интуитивный синтаксис; кроссплатформенность и другие. Все это не относится к методике, но является важным фактором, который влияет на выбор языка.

Чтобы обосновать требования к языку программирования, необходимо, в первую очередь, определить цель изучения программирования.

К сожалению, наличие четких целей не дает нам единого ответа на вопрос, какой язык программирования лучше изучать со школьниками, так как многие из современных и не только языков программирования я легкостью удовлетворяют каждый из указанных целевых аспектов.

Проанализировав ФГОС, авторские методики и мнение учителей информатики, в ходе исследования были выдвинуты следующие требования:

- язык программирования, выбранный для обучения, должен быть максимально приближен к алгоритмическому языку по своей структуре, так как он является наиболее легким для усвоения принципов программирования учащимися (УМК И.Г. Семакина);

- выбранный язык программирования должен поддерживать структурную и объектно-ориентированную парадигму программирования (УМК Н.Д. Угриновича);

- код, составленный на выбранном языке программирования, должен быть легко читаем;

- язык должен быть строго типизированным, ибо смешение целых чисел, вещественных чисел и текстовых переменных приводит у начинающих программистов к неправильному представлению о методах хранения данных в памяти компьютера [9];

- язык программирования должен иметь возможности: вычисления всех элементарных математических функций; выполнения всех арифметических операций ("+", "-", "*", "/"); реализации: логических операций сравнения: ">", ">=", "=", "<>", "<=", "<"; логических операций:

"И", "ИЛИ", "НЕ"; наличие аналога логической функции: ЕСЛИ:ТО:ИНАЧЕ.; наличие оператора, позволяющего реализовать цикл или хотя бы оператор безусловного перехода; выбранный язык программирования должен предоставлять возможность организации массивов: в перечне возможных заданий ЕГЭ указаны задачи на обработку данных конечных числовых последовательностей (массивов, списков)

Таким образом, в ходе сравнения языков программирования, выявили самый подходящий для глубокого изучения в школе – Python. На нём можно разработать что угодно, огромное количество библиотек и возможность писать простой и понятный код.

1.3 Особенности обучению программирования в курсе информатики средней школы

Несмотря на единые требования образовательного стандарта, у каждого учителя свой набор требований к языку программирования. Это может быть удобная, дружелюбная среда разработки; простой, интуитивный синтаксис; кроссплатформенность и другие. Все это не относится к методике, но является важным фактором, который влияет на выбор языка.

Чтобы обосновать требования к языку программирования, необходимо, в первую очередь, определить цель изучения программирования.

В. Г. Есауленко утверждает, что к сожалению, наличие четких целей не дает педагогам единого ответа на вопрос, какой язык программирования лучше изучать со школьниками, так как многие из современных и не только языков программирования с легкостью удовлетворяют каждый из указанных целевых аспектов [11, с. 48].

Исходя из этого, на сегодняшний день существует большое число исследований, методического плана, на предмет формирования у учащихся основной школы основ алгоритмизации и программирования, как

компетенций необходимых конкурентоспособному специалисту в области информатизации общества и производства.

Сегодняшние школьные учебники по информатике, чаще всего разбит на определенные разделы. Все эти разделы непрерывно связаны с изучением как основ предмета информатики, так и обучением информационно-коммуникационных технологий. В большинстве из них тема «Алгоритмизация и программирование» рассматривается хаотично и не полностью. Более того, как отмечает Д. П. Кириенко, некоторые методические пособия предлагают разные количество часов для изучения этой темы, а материал может представлять собой сложную структуру, что даёт предпосылки для не полного освоения материала. В связи с эти перед началом изучения в школе учителю необходимо разработать и внедрить методический материал, который максимально упростит обучение учеников программированию. В то же время учитель использует свои уроки личного развития, использует собственный опыт, полученный ранее, и применяет учебные пособия, которые в более простой форме содержат материал для изучения выбранного языка программирования [14].

Изучая Федеральный Государственный Стандарт основного общего образования, в разделе «11.3. Математика и информатика» наиболее применимые к школьной дисциплине Информатике это пункты 10 и 12-14. В этих пунктах основное внимание уделяется алгоритмизации и алгоритмическому мышлению, а также умению работать с информацией [43].

Алгоритмическое и логическое мышление можно развивать в рамках изучения темы «Алгоритмизация и программирование». Умению работать с информацией в целом, посвящен практически весь курс информатики, но практическому применению этого навыка уделяется недостаточно внимания, как и теме «Алгоритмизация и программирование».

ФГОС ООО отводит 105 часов на изучение информатики: 7-9 классы исходя из расчета 35-и академических часов на год, при одном уроке в

неделю. Тема «Алгоритмизация и программирование» по плану изучается 19 часов и при этом планируется, что учащиеся будут знать программирование на базовом уровне [43].

Предоставленный объем часов на изучение темы «Алгоритмизация и программирование» в рамках предмета информатика небольшой, но при этом объем рассматриваемого материала достаточно объемен.

В ходе работы были проанализированы алгоритмические разделы в учебно-методических комплексах (далее: УМК) по курсу информатика И. Г. Семакина, Л. Л. Босовой, Н. Д. Угриновича, К. Ю. Полякова. Рассмотрим подробнее каждый из них.

Н. Д. Угринович начинает рассмотрение раздела алгоритмизации только в девятом классе, при этом изучение основ программирования происходит на языке Visual Basic. В учебнике Н. Д. Угриновича затрагиваются такие темы как определение алгоритма и его формальное исполнение; тип, значение и имя переменных; алгоритмические структуры и кодирование их основных типов с использованием языков процедурного и объектно-ориентированного программирования; функции и графические возможности с использованием языков процедурного и объектноориентированного программирования; логические, арифметические и строковые выражения [42].

И. Г. Семакин, также, как и Н. Д. Угринович начинает обучение раздела алгоритмизации в девятом классе, но при этом делит его на две разные главы. Первая глава посвящена управлению алгоритмами и включает в себя такие темы как: кибернетика и управление с обратной связью; свойства алгоритма и его определение; виды алгоритмов, такие как прямой, ветвление и циклический; учебный исполнитель в графической интерпретации. Во второй главе рассказано о программирование на языке Pascal и рассматриваются темы: понятие программирование; знакомство с языком Pascal; работа с ранее изученными видами алгоритмов: линейное, ветвящееся, циклические; составление диалога с компьютером; работа с

таблицами и массивами; поиск экстремумов массива и способы его сортировки [36].

Базовый уровень изучения информатики в старших классах. В качестве основного языка программирования, изучаемого на уроке выбран язык Pascal. Обучение программированию строится по «стандартной» схеме. Сперва изучается структура программы, основные типы данных. Изучаются конструкции языка: линейные алгоритмы, разветвляющиеся алгоритмы, циклы с предусловием, параметром (а также вложенные циклы). Элементы языка, типы данных: числовые (целые и дробные), логические, символьные, комбинированные. Структуры данных: переменные, массивы. При изучении массивов, особое внимание уделяется типовым алгоритмам обработки оных. Вспомогательные алгоритмы и подпрограммы – как важная часть курса. Однако, изучение программирования в рамках данной программы, проходит только в 10 классе. В 11 классе разделу «Алгоритмизация и программирование» не выделено часов [37;38].

Л. Л. Босова, также, как и И. Г. Семакин, делит этот раздел на два подраздела и для обучения использует язык программирования Pascal, но в отличие от предыдущих авторов, она начинает обучение алгоритмизации в восьмом классе. Основные темы, рассматриваемые Л. Л. Босовой в первой главе: что такое алгоритм, его определение, исполнители и свойства; различные формы записи алгоритмов, такие как: словесные блок-схемы и алгоритмические языки. Изучая различные объекты алгоритмов, автор затрагивает такие темы как: величины, выражения, команды присвоения и табличные величины. Далее рассматриваются такие алгоритмические конструкции как: следование, ветвление и повторение. Во второй главе Людмила Леонидовна рассматривает такие темы как: общие сведения о языке, его алфавит, типы данных, структура и оператор присвоения; ввод и вывод данных; ввод данных с клавиатуры; программирование различных типов данных таких как: числовые, целочисленные, символьные и логические; программирование ранее изученных типов алгоритмов, таких

как: линейные, разветвляющиеся и циклические. Автор использует следующие варианты программирования цикла: цикл с постусловием цикл, с предусловием и цикл с заданным количеством повторений [5;6;7;8].

В отличие от предыдущих авторов, К. Ю. Поляков начинает рассмотрение алгоритмизации ещё в седьмом классе. Кирилл Юрьевич затрагивает следующие темы в первой главе: алгоритмы и их исполнители; способы и формы их записи; вспомогательные алгоритмы; введение переменных; графические примитивы и анимация; управление алгоритмами с помощью клавиатуры; работа с линейными, разветвляющимися и циклическими алгоритмами с условием [7]. В восьмом классе К. Ю. Поляков накладывает новую информацию на ранее изученную, тем самым освежая ранее полученные знания и углубляя их за счёт введения в программирование. Автор разбирает виды алгоритмов, виды циклов и работу с массивами [28]. В девятом классе К. Ю. Поляков расширяет ранее полученные знания по программированию следующими темами: обработка текста и массивов; матрица; сложные алгоритмы; работа с процедурами и функциями [29].

В 10-11 классе на изучение программирования отводится 101 час на раздел Алгоритмы и программирование. Учебник Константина Полякова рассчитан на профильные классы и обеспечивает углубленное изучение программирования. Изучаются различные принципы и подходы к программированию (процедурное и динамическое), алгоритмы и структуры данных (списки, массивы, множества, стеки, деревья, графы), а также работа с файлами. Языки программирования, которые изучаются в рамках курса – Python и C++, что выделяет этот учебный курс, относительно других (где изучается язык Pascal) [25;26;30].

Проанализируем учебник по информатика (10-11 класс) под авторством А. Г. Гейн, Н. А. Юнерман Базовый уровень. Изучение программирования начинается с понятия и свойств алгоритма. Далее идет переход на алгоритмы для исполнителей. На примере исполнителя

«Паркетчик» идет объяснение базовых алгоритмических структур: ветвления, циклы, вспомогательные алгоритмы. После «Паркетчика» следует обзор других исполнителей. На работу «С настоящим языком программирования» отводится всего 2 часа. На кратком обзоре двух языков программирования высокого уровня (Basic и Pascal) в принципе заканчивается изучение программирования [9].

Рассмотрим учебник информатики (10-11 класс) под авторством А. Г. Гейн, А. Б. Ливчак, А. И. Сенокосов и др. Учебная программа предполагает изучение информатики в 10-11 классах на углубленном уровне. Программа рассчитана на 280 часов (из расчета 4 в неделю в профильном классе). Упора на какой-то конкретный язык программирования высокого уровня авторы не делают – тут умения учителя. Изучение программирования начинается с понятия «алгоритм» и его свойства, далее идут алгоритмы для исполнителей (Машина Тьюринга). Изучение языка программирования начинается с понятия переменной и типов данных. Вводятся понятия вспомогательный алгоритм и пошаговая детализация; изучаются массивы, циклы; рекуррентным соотношениям и рекурсивным алгоритмам уделяется особое внимание в рамках этой программы. Алгоритмы обработки массивов (в 10 классе). В 11 классе алгоритмы изучаются только с точки зрения математической теории алгоритмов, т.е., учащиеся не используют язык программирования [12].

Следующим рассмотрим учебник информатики (10-11 класс) под ред. Н. В. Макаровой. Рабочая программа изучения информатики на базовом уровне. Как основной язык программирования выбран Pascal (также возможно использовать Basic). Сперва изучается понятие «алгоритм», затем идут способы записи алгоритма. Постепенно осуществляется переход к языку программирования высокого уровня. В 10-11 классе изучаются основные алгоритмические конструкции: линейные алгоритмы, алгоритмы ветвления, циклические алгоритмы. Массивы и числовые переменные (целые и дробные) как основные структуры данных. В рамках базового

курса изучаются алгоритмы обработки чисел, потока данных, а также типовые алгоритмы обработки целочисленных одномерных массивов. Особое внимание уделяется задачам на проведение ручной трассировки программы. Например, дана определенная программа, написанная на языке высокого уровня, даны входные данные и учащемуся необходимо определить какой результат будет на выходе [13].

Можно сделать вывод, что у разных разработчиков УМК по курсу «Информатика» самые разные мнения о том, каким образом должна быть построена содержательная линия раздела алгоритмизации и программирования. Следует отметить, что Н. Д. Угринович, И. Г. Семакин и Л. Л. Босова при написании своих УМК придерживаются линейной модели обучения, в то время как К. Ю. Поляков использует спиральную модель обучения Джерома Брунера

Касаемо выбора первого языка для изучения программирования, видим, что Н. Д. Угринович использует Visual Basic, а Л. Л. Босова, И. Г. Семакин и К. Ю. Поляков ориентируются на Pascal.

Как справедливо отмечает, О. А. Кочеткова, Ю. Н. Пудовкина – раздел «Программирование и основы алгоритмизации» позволяет показать важность и других изучаемых разделов, умений и навыков. На конкретных примерах можно показать, как программирование, создание игр и проектов объединяет: виды информации, информационные процессы, способы представления информации, математическую логику, кодирование информации, навыки работы с текстовой и звуковой информации, навыки работы с компьютерной графикой, программное и аппаратное обеспечение компьютера [16].

Помимо основной программы обучения в школе, как считает О. А. Кочеткова, И. В. Долгополов, важным элементом приобретения новых компетенций являются элективные курсы. Внедряя в школах элективные курсы, следует помнить, что речь идет о всей методической системе обучения этими курсами в целом. В современных условиях

педагоги все чаще стараются разрабатывать межпредметные элективные курсы, содержание которых бы пересекалась между большими смежными дисциплинами. Посещение таких занятий позволяет ученикам выбрать именно ту область знаний и наук, которая подходит индивидуально им, а также углублять в них свои теоретические и практические знания и интересы [15, с. 104].

А. В. Душкин считает, что элективные курсы предназначены для перехода к профильному образованию, являясь элементом учебного плана, они дополняют содержание курса и помогают выстраивать профильное обучение. Данная форма позволяет удовлетворить интересы обучающихся различной направленности, что позволяет индивидуализировать образовательный процесс, так как тематика таких курсов может лежать как в пределах общеобразовательной программы, так и вне ее. Также элективные курсы являются занятиями по выбору, что позволяет обучающимся выбрать ту или иную интересующую его тематику, таким образом определяя свой профессиональный путь [10, с. 101].

Выстраивая курс необходимо учитывать, что наибольшую эффективность можно достичь лишь при помощи активных форм организации занятий. Также проектные формы работы могут повлиять на мотивацию к изучению курса обучающимися.

При изучении языка программирования Python, в рамках элективного курса, учащиеся с легкостью смогут освоить синтаксис, в котором используются минимальное количество вспомогательных слов. Уже через несколько занятий смогут писать отдельные модули программ для дальнейшего внедрения и будут развивать способности, без которых в будущем сложно будет представить современное общество. В свою очередь учащиеся, которые хотят далее связать свою профессию с информационными технологиями смогут успешно подготовиться к экзаменам, так как все задачи, связанные с программированием, включают также описания на языке Python.

Выводы по главе 1

В теоретической части рассмотрено понятие языка программирования, несколько классификаций языков программирования, сравнили наиболее популярные из них. Определили основные критерии, по которым выбирается язык программирования для изучения в школе. Также были рассмотрены учебные материалы, существующие для школ и особенности включения курса «программирование» в школьную программу предмета информатика.

Обоснована целесообразность элективного курса «Основы программирования на языке Python», как наиболее подходящего, на наш взгляд, способа более детального изучения учащимися основной школы курса «Алгоритмизация и программирование», а также способа привлечения большего интереса учащихся к предмету «Информатика».

Таким образом, учитывая особенности результатов и требований ФГОС, язык программирования Python является наиболее оптимальным решением. К тому же, мотивация к изучению «устаревших» языков программирования, по мнению обучающихся, намного ниже, чем мотивация к изучению современных языков программирования с их возможностями, которые они предоставляют, облегчая процесс создания программ и делая его увлекательнее. Поэтому заинтересованные ученики наиболее охотно будут изучать молодой и перспективный Python, нежели Pascal. Для незаинтересованных в обучении программированию, Python максимально облегчит работу массой встроенных функций и простотой синтаксиса.

ГЛАВА 2. МЕТОДИЧЕСКИЕ ОСОБЕННОСТИ ИЗУЧЕНИЯ ЯЗЫКОВ ПРОГРАММИРОВАНИЯ В ШКОЛЕ

2.1 Элективный курс «Основы программирования на Python»

Курс направлен на изучение языка программирования Python и практической реализации знаний.

В качестве языка программирования был выбран язык программирования Python по ряду причин: это современный быстроразвивающийся язык. В школьной информатике в последнее время он стал занимать особое место, многие учебники и образовательные ресурсы выбирают Python как основной язык программирования, современные задачки по информатике в разделе «Программирование» предполагают решение задач по программированию на языке Python; Синтаксис Python довольно простой, что обеспечивает быстрое изучение и усвоение языка [14]. Для того, чтобы писать программу на языке Python достаточно всего лишь установить Python на компьютер/телефон/планшет и писать программы, без дополнительной настройки среды [14].

Компактные конструкции языка делают программы на Python намного короче относительно других языков таких как C++ и Pascal. Это позволяет писать небольшие по объему программы, что в условиях ЕГЭ является важным фактором: при написании программ, учащиеся пишут намного меньше (что в условиях ограниченности времени на ЕГЭ будет смотреться более выигрышно);

Изучать работу функций и основных алгоритмов на Python объективно проще.

Python – это полноценный и многофункциональный язык программирования. Этот язык применяется в самых различных сферах: начиная от web-программирования и заканчивая программированием микроконтроллеров [14].

Ученики, которые сдают ЕГЭ по информатике в основном планируют связать свою жизнь с IT. На данный момент язык программирования Python сейчас востребован во многих предприятиях, особенно это выражено в Москве и Санкт-Петербурге. Таким образом изучение языка в школе может способствовать формированию базиса для дальнейшего более углубленного освоения в высшем учебном заведении [10].

При разработке методических рекомендаций по организации элективного курса при изучении программирования с использованием Python опираемся на содержание программы разработанной нами на основе примерной программы по информатике в 7-9 классах, рекомендованной ФГОС.

Адрес сайта: <https://sites.google.com/view/language-programs/главная>

Цель курса: знакомство с основами программирования на языке Python.

Задачи курса:

- сформировать у учащихся интерес к профессиям, связанным с программированием;
- создание условий формирования алгоритмической культуры учащихся;
- развитие алгоритмического мышления учащихся;
- формирование базовых понятий структурного программирования;
- приобрести знания и навыки алгоритмизации в ее структурном варианте;
- освоение учащимися методов решения задач, реализуемых на языке Python.

Рассмотрим требования к уровню освоения содержания дисциплины. Учащиеся должны знать основные элементы языка программирования, его основные приемы и алгоритмы программирования, основные численные методы решения задач. Учащиеся должны уметь разрабатывать алгоритмы

необходимые для решения поставленных задач, разрабатывать алгоритмы, используя основные приемы программирования, а также проводить отладку и тестирование программы, проводить необходимые расчеты на ПК.

Курс рассчитан на 11 часов.

Проведение контрольных мероприятий и оценивание работ не предусмотрено в связи с ознакомительной направленностью курса.

Контент данного курса соответствует следующим основным требованиям: не противоречит требованиям действующих образовательных стандартов, содержание курса соответствует современному научному и учебно-методическому уровню соответствующей области знаний, изложенный учебный материал последователен, логичен и полон, используются контрольно-оценочные средства для оценивания и самоконтроля уровня освоения учебного материала. Имеет возможность модернизации в процессе его применения в учебном процессе.

Системный подход реализуется за счет рассмотрения элективного курса, как единого целого, то есть каждый урок и каждое задания связаны в единую систему. Также в курсе присутствует иерархичность заданий, заключающаяся в том, что весь материал выстроен таким образом, чтобы каждое последующее занятие основывалось на знаниях, полученных из прошлого урока. Совокупность всех компонентов данного подхода позволяет рассмотреть учебные задачи в тесной взаимосвязи со средствами и методами обучения программированию.

Деятельностный подход заключается в поддержании мотивации обучающегося к обучению за счет активного и эффективного применения информационно-коммуникационных технологий. Наряду с этим, в курсе присутствует творческая направленность, позволяющая учащимся самостоятельно решать поставленные учителем задачи. Данный элективный курс также направлен на развитие инженерного мышления и будущих профессиональных взглядов.

Системный и деятельностный подход в совокупности стимулируют мотивацию обучающихся, тем самым способствуют усвоению знаний. Системно-деятельностный подход также реализует идеи непрерывного образования и дает возможность способствовать формированию универсальных учебных действий.

В качестве методов обучения используются: словесный метод; проблемный метод; исследовательский метод; проектный метод.

Словесный метод используется на протяжении всего элективного курса, включает в себя: лекции, объяснение заданий, рассказ.

Проблемный метод используется для решения поставленных задач, основанных на уже имеющихся знаниях и сформированных навыках обучающихся. Данный метод используется для того чтобы у учащихся возникало осознание того, что им не хватает имеющихся знаний и для достижения конечного результата им нужно найти ответ самостоятельно путем поиска новых знаний.

Исследовательский метод способствует овладению элементами научного познания: осознанию проблемы, выдвижению гипотез, построению плана ее проверки и умение делать выводы.

Проектный метод используется для развития творческих и познавательных способностей, критического мышления, умения самостоятельно получать знания и применять их на практике.

Перечисленные методы используются в совокупности на протяжении всего элективного курса, что способствует эффективному построению занятий

Таким образом, так как при изучении разделов информатики, относящихся к теме «Алгоритмизация и программирование» отводится недостаточное количество времени для практического применения знаний, элективный курс может не только решить данную проблему, но и расширить знания учащихся по данным разделам. Также одной из целей реализации курса выступает дальнейшая самореализация и самоопределение учащихся

по отношению к дальнейшему обучению. К моменту проведения элективного курса учащиеся должны знать: базовые алгоритмические структуры; запись алгоритма в виде блок-схем.

Комплексно-тематическое планирование элективного курса представлено в таблице 4.

Таблица 4 – Комплексно-тематическое планирование элективного курса

№ п/п	Тема	Теория	Практика
1.	Введение в программирование	0	0
2.	Введение в язык программирования Python	1	1
3.	Типы данных. Определение переменной	1	1
4.	Логические выражения	1	1
5.	Условный оператор. Инструкция if	1	0
6.	Цикл While	1	0
7.	Ввод данных с клавиатуры	1	1
8.	Последовательности: строки	1	1
9.	Последовательности: списки	1	1
10.	Решение задач	2	10
Итого:		10	16

2.2 Методическое содержание элективного курса «Основы программирования на Python»

С целью наглядности был разработан сайт элективного курса (рис.1).

В котором мы кратко представили современные языки программирования, а также разместили рубрики: «Материалы для учителя» и «Материалы для ученика».

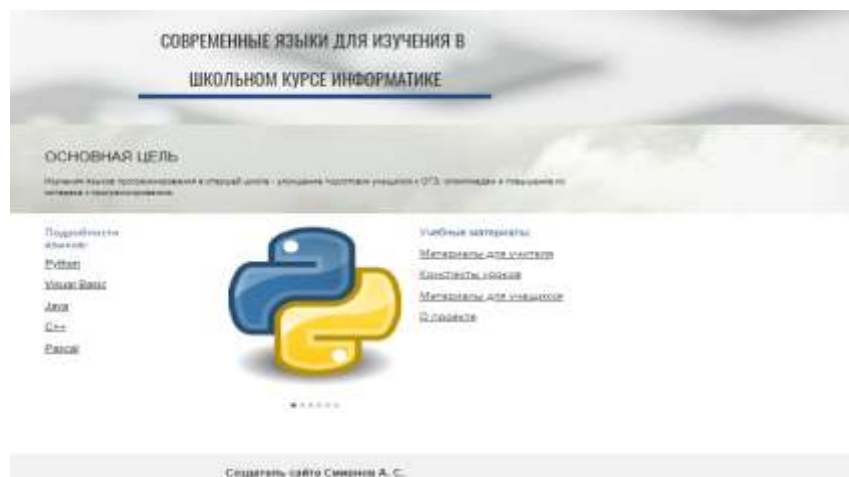


Рисунок 1 – Главная страница

В данном параграфе представим методические рекомендации по темам, предлагаемым для изучения в рамках элективного курса.

Занятие 1. Введение в программирование

Тип урока: урок усвоения новых знаний.

Задачи:

Образовательная: сформировать представление о языках программирования.

Развивающая: развивать у учащихся логическое и алгоритмическое мышление, навыки мыслительной деятельности, включая каждого учащегося в учебно-познавательный процесс и создавая условия для работы каждого в индивидуальном психологическом ритме.

Воспитательная: воспитывать у учащихся активность, интерес к предмету.

Цель: знакомство с историей возникновения языков программирования, разница между машинным и высокоуровневым языками, введение в языки программирования, введение в разработку ПО, знакомство с компилятором.

Занятие 2. Введение в язык программирования Python

Тип урока: урок усвоения новых знаний.

Задачи урока:

Цель: знакомство с языком программирования Python, изучить основы программирования на данном языке.

Задачи:

Образовательная: сформировать представление о языке Python, научиться писать простой код на языке программирования Python в среде разработки.

Развивающая: развивать у учащихся логическое и алгоритмическое мышление, навыки мыслительной деятельности, включая каждого учащегося в учебно-познавательный процесс и создавая условия для работы каждого в индивидуальном психологическом ритме.

Воспитательная: воспитывать у учащихся самостоятельность, активность, интерес к предмету.

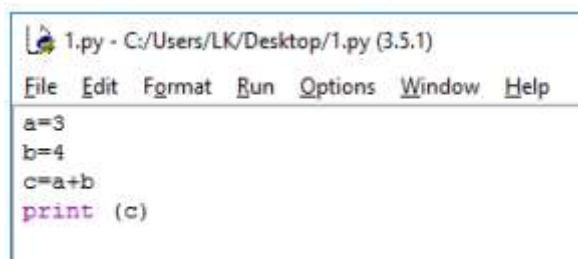
Непосредственно для того, чтобы начать обучение языку необходимо установить с официального сайта <https://www.python.org/downloads/> программное средство, включающее в себя среду разработки IDLE.

После запуска среды разработки можно работать в интерактивном режиме, что означает при написании команды интерпретатор тут же выполняет ее. Интерактивный режим позволяет тут же проверять правильность написания кода, не компилируя его. Начинать изучение языка лучше всего с элементарных примеров, основанных на использовании математических выражений (рис. 2).

```
>>> 2+2
4
>>> 5*2
10
>>> 5**2
25
>>> 10/2
5.0
>>>
```

Рисунок 2 – Пример использования математических выражений

Необходимо пояснить, что при использовании операции возведения в степень необходимо использовать два символа, умножения. Написание остальных основных математических операций понятно на интуитивном уровне. При необходимости написать полный код программы для последующего его выполнения необходимо создать новое окно документа, написать код (рис. 2) и выполнить его с помощью команды Run Module или нажатии клавиши F5, после чего в среде разработки будет выведен результат выполнения модуля:



```
1.py - C:/Users/LK/Desktop/1.py (3.5.1)
File Edit Format Run Options Window Help
a=3
b=4
c=a+b
print (c)
```

Рисунок 3 – Код модуля

Ответ будет выведен отдельно (рис. 4):

```
===== RESTART: C:/Users/LK/Desktop/1.py =====  
7  
>>>
```

Рисунок 4 – Пример ответа

Файлы модулей сохраняются с расширением «.py». Также существуют настройки, позволяющие выполнять модуль двойным нажатием на файл с написанным кодом. Вообще код программы можно писать и обычном в текстовом документе, но для удобства проверки кода лучше использовать среду разработки.

Занятие 3. Типы данных. Определение переменной

Тип урока: урок усвоения новых знаний.

Цель урока: познакомить учащихся с основными понятиями.

Задачи урока:

Образовательная: познакомить учащихся с основными понятиями, научить применять на практике.

Развивающая: развивать у учащихся логическое и алгоритмическое мышление, навыки мыслительной деятельности, включая каждого учащегося в учебно-познавательный процесс и создавая условия для работы каждого в индивидуальном психологическом ритме.

Воспитательная: воспитывать у учащихся самостоятельность, активность, интерес к предмету.

В основном компьютерная программа оперирует с числовыми и текстовыми типами данных. На прошлом занятии рассматривались такие операции над числовыми типами данных, как сложение, умножение, возведение в степень. Но нужно понимать, что типы числовых данных могут отличаться, числа могут быть целыми, дробными или очень большими.

Основные типы данных, над которыми мы будем производить операции:

– целые числа (integer) – положительные и отрицательные целые числа, включающие 0 (например 3, - 6, 0, 257);

- числа с плавающей точкой (float point) – дробные числа (например, 1.55, 3.789654, -0.0523). Стоит заметить, что знаком разделения целой и дробной части является точка, а не запятая;

- строки (string) — набор символов, заключенных в кавычки (например, «Apple», «Лист», 'lg5', '959'). Кавычки могут быть одинарными или двойными. В последних версиях среды разработки IDLE существует поддержка кириллицы.

Над данными (операндами) можно производить действия (операции). Для выполнения действий нужны специальные инструменты - операторы (рис. 5).



Рисунок 5 – Оператор сложения

Запрограммировано, что оператор «+» выполняет операцию сложения над операндами, тип данных которых integer, то есть складывается два целых числа. Если же применить ту же операцию к операндам типа string, то есть к двум строкам, то результатом выполнения будет соединение двух строк, например: «Меня» + «зовут» + «Максим»

Результатом выполнения операции будет являться строка: «Меня зовут Максим».

Также в языке Python можно выполнять операции над не однотипными типами данных, например, при сложении целого и вещественного числа получится вещественное (34 + 12,5 = 49,5). Но при соединении таких типов данных, как целое число и строка будет выводиться ошибка.

Бывают случаи, когда результат выполнения операции нужно вывести в определенном виде, например, перевести целое число в строку, тогда нужно использовать функцию str(), которая переводит из числового типа

данных в текстовый. Таким же образом функция `int()` преобразует в целочисленный тип данных, а функция `float()` в вещественный тип данных (табл. 5).

Таблица 5 – Типы данных

Выражение	Результат выполнения
<code>Int(2.04)</code>	2
<code>Str(74)</code>	“74”
<code>Float(37)</code>	37.0

Чтобы связать данные с именем необходимо произвести операцию присваивания данных к переменной – зарезервированной области памяти, которая обозначается символом «`=`» (например, `V = 5`), таким образом означает, что данные в определенной области памяти связаны с именем `V` и обращаться к ним теперь следует по этому имени (рис. 6).



Рисунок 6 – Оператор присваивания

Занятие 4. Логические выражения.

Тип урока: урок усвоения новых знаний.

Цель: знакомство учащихся понятиями логическое выражение и логический тип данных, а также с логическими операторами; научиться составлять сложные логические выражения; создание программы на языке Python.

Задачи:

Образовательная: сформировать представление о логических выражениях.

Развивающая: развивать у учащихся логическое и алгоритмическое мышление, навыки мыслительной деятельности, включая каждого

учащегося в учебно-познавательный процесс и создавая условия для работы каждого в индивидуальном психологическом ритме.

Воспитательная: воспитывать у учащихся активность, интерес к предмету.

Зачастую в реальной жизни нам необходимо согласиться или отрицать то или иное событие, высказывание. Например, «Сумма чисел 6 и 5 больше 7» является правдой или истинным выражением, «Сумма чисел 9 и 4 меньше 7» – ложным. Можно заметить, что эти фразы предполагают только два ответа: «Да» (правда) и «Нет» (ложь). Подобное есть и в программировании: если результатом вычисления выражения является «истина» или «ложь», то такое выражение называется логическим. Помимо целых, вещественных и строковых типов данных также выделяют логический тип данных. У этого типа всего два возможных значения: True (правда) – 1 и False (ложь) – 0. Эти значения и являются результатом логических выражений.

Чтобы получить истину (True) при использовании оператора конъюнкции and, необходимо, чтобы результат обоих простых выражений были истинными. Если хотя бы в одном случае результатом будет False (ложь), то и все сложное выражение будет ложным. Чтобы получить истину (True) при использовании оператора or, нужно, чтобы результат хотя бы одного простого выражения, входящего в состав сложного, был истинным. В данном случае сложное выражение становится ложным лишь тогда, когда ложны все, входящие в него простые выражения.

Задание 5. Условный оператор. Инструкция if

Тип урока: урок усвоения новых знаний.

Цель урока: познакомить учащихся с условным оператором, инструкцией if-elif-else.

Задачи урока:

Образовательная: познакомить учащихся с основными понятиями, научить применять на практике.

Развивающая: развивать у учащихся логическое и алгоритмическое мышление, навыки мыслительной деятельности, включая каждого учащегося в учебно-познавательный процесс и создавая условия для работы каждого в индивидуальном психологическом ритме.

Воспитательная: воспитывать у учащихся самостоятельность, активность, интерес к предмету.

Условная инструкция if-elif-else (оператор ветвления) – основной инструмент выбора в Python. Говоря простым языком, она выбирает, какое действие следует выполнить, в зависимости от значения переменных в момент проверки условия. Существует несколько вариаций использования данной инструкции.

1. Условная конструкция if

Команда if в Python работает по той же схеме, что и в других языках программирования. Она содержит в себе логическое условие, и если это условие истинно (равно True) - выполнится блок кода, записанный внутри команды if. Если же логическое условие ложно (равно False), то блок кода записанный внутри команды if пропускается, а выполнение кода переходит на следующую после блока if строчку кода (рис. 7).

```
>>> a=10
>>> b=2
>>> if a<100:
        c=a*b

>>> c
20
```

Рисунок 7 – Пример использования оператора if

В конструкцию if может быть добавлена команда else. Она содержит блок кода, который выполняется, если условие в команде if ложно (рис. 8).


```

>>> t1=30
>>> t2=50
>>> if t1+12>99:
        print ("Сумма больше 99")
else:
        print ("Сумма меньше 99")
Сумма меньше 99

```

Рисунок 8 – Пример кода с веткой else

Команда else является опциональной, в каждой if - конструкции может быть только одна команда else.

Команда elif позволяет проверить истинность нескольких выражений и в зависимости от результата проверки выполнить нужный блок кода.

Как и команда else, команда elif является опциональной, однако, в отличие от команды else, у одной if-конструкции может существовать произвольное количество команд elif (рис. 9).

```

>>> x=-10
>>> if x>0:
        print (1)
elif x<0:
        print (-1)
else:
        print (0)
-1

```

Рисунок 9 – Запись программы с использованием инструкции if-elif-else

Занятие 6. Цикл While.

Тип урока: урок усвоения новых знаний.

Цель урока: познакомить учащихся с Циклом While.

Задачи урока:

Образовательная: познакомить учащихся с Циклом While, научить применять на практике.

Развивающая: развивать у учащихся логическое и алгоритмическое мышление, навыки мыслительной деятельности, включая каждого

учащегося в учебно-познавательный процесс и создавая условия для работы каждого в индивидуальном психологическом ритме.

Воспитательная: воспитывать у учащихся самостоятельность, активность, интерес к предмету.

Циклы While - это инструкции, позволяющие выполнять одну и ту же последовательность действий (блок), пока действует определенное условие.

В повседневной жизни мы очень часто встречаемся с циклами. Выполняя одни и те же действия пока какое-либо условие истинно. Например, нам нужно заполнить ящик, пока в нем есть место мы будем его заполнять, как только места не осталось мы останавливаемся.

Универсальным организатором цикла в языке программирования Python (как и во многих других) является инструкция while. Слово «while» с английского языка переводится как «пока» («пока действует что-то, делать то-то»).

Конструкцию инструкции while на языке Python можно описать так:

While a логический_оператор b:

Выражения, выполняемые при результате

True в логическом выражении

Изменение a

Эта схема приближительна, т.к. логическое выражение в заголовке цикла while может быть более сложным, а изменяться может переменная (или выражение) b.

Может возникнуть вопрос: «Зачем изменять a или b?». Когда выполнение программного кода доходит до цикла while, выполняется логическое выражение в заголовке и, если было получено True (истина), выполняются вложенные выражения. После, поток выполнения кода программы снова возвращается в начало цикла while, и снова проверяется условие. В случае, когда условие всегда будет истинным, то тело цикла будет повторяться бесконечное количество раз.

Чтобы этого не произошло, нужно организовать момент выхода из цикла – сделать так, чтобы выражение в какой-то момент стало ложным. Так, например, изменяя значение какой-либо переменной в теле цикла, логическое выражение может стать ложным (рис. 10).



Рисунок 10 – Пример блок-схемы ложного выражения

Эту изменяемую переменную, доводящую условие до ложности, обычно называют счетчиком. Как и всякой переменной ей можно давать произвольные имена, однако очень часто используют букву *i*. Простейший цикл может выглядеть так (рис. 11).

```
>>> str1="+"
>>> i=0
>>> while i < 10:
    print(str1)
    i=i+1
```

Рисунок 11 – Пример простейшего цикла

В последней строчке данного скрипта происходит увеличение значения переменной *i* на единицу, поэтому с каждым проходом цикла она на нее и увеличивается. Когда будет достигнуто значение 10, логическое выражение $i < 10$ даст ложный результат, выполнение тела цикла будет остановлено, а поток выполнения программы перейдет на команды

следующие за всей конструкцией цикла. Результатом выполнения будет вывод на экран десяти знаков + в столбик.

Занятие 7. Ввод данных с клавиатуры

Тип урока: урок усвоения новых знаний.

Цель урока: познакомить учащихся с вводом данных с клавиатуры.

Задачи урока:

Образовательная: познакомить учащихся с основными понятиями, научить применять на практике.

Развивающая: развивать у учащихся логическое и алгоритмическое мышление, навыки мыслительной деятельности, включая каждого учащегося в учебно-познавательный процесс и создавая условия для работы каждого в индивидуальном психологическом ритме.

Воспитательная: воспитывать у учащихся самостоятельность, активность, интерес к предмету.

Для удобного обращения с программой пользователю предусмотрен ввод и вывод данных. За вывод данных отвечает функция `print()`, которая уже была использована в предыдущих темах.

Ввод же данных с клавиатуры осуществляется с помощью функции `input()`. Если при выполнении выражения встречается данная функция, то выполнение приостанавливается в ожидании ввода данных с клавиатуры. После ввода данных и нажатия клавиши `Enter` функция завершается, а введенные данные могут быть использованы в дальнейшем выполнении кода (рис. 12).

```
>>> input ()
37
'37'
>>> input ()
Hello, its me
'Hello, its me'
```

Рисунок 12 – Ввод данных с помощью функции `input()`

Выполняющаяся программа может запросить ввод данных, хотя пользователь не всегда может определить для чего это нужно. Для решения этой проблемы можно использовать параметр-приглашение, тогда при запросе ввода данных программа будет выводить соответствующее сообщение, записанное в кавычках (рис. 13).

```
>>> input("Введите свой возраст: ")
Введите свой возраст: 20
'20'
>>> input("Введите имя: ")
Введите имя: Максим
'Максим'
```

Рисунок 13 – Параметр-приглашение

Данные, введенные таким образом всегда возвращаются в виде строки, чтобы получить число необходимо использовать функции `int (input())` и по аналогии `float (input())` (рис. 14).

```
>>> int (input("Введите число: "))
Введите число: 74
74
>>> float (input("Введите число: "))
Введите число: 47
47.0
```

Рисунок 14 – Пример использования функций `int (input())` и `float (input())`

Также результат, возвращаемый функцией `input()` может быть присвоен переменной для дальнейшего ее использования в программе (рис. 15).

```
>>> Имя = input ("Введите свое имя: ")
Введите свое имя: Максим
>>> Имя
'Максим'
```

Рисунок 15 – Пример присвоения переменной результата, возвращаемого функцией `input()`

Урок 8. Последовательности: строки

Тип урока: урок усвоения новых знаний.

Цель урока: познакомить учащихся с основными понятиями: «строки».

Задачи урока:

Образовательная: познакомить учащихся с основными понятиями, научить применять на практике.

Развивающая: развивать у учащихся логическое и алгоритмическое мышление, навыки мыслительной деятельности, включая каждого учащегося в учебно-познавательный процесс и создавая условия для работы каждого в индивидуальном психологическом ритме.

Воспитательная: воспитывать у учащихся самостоятельность, активность, интерес к предмету.

Строка – последовательность, упорядоченный набор данных одного типа (букв или символов)

Строки в языке Python могут быть заключены как в одиночные, так и в двойные кавычки, однако начало и конец строки должны обрамляться одинаковым типом кавычек.

Также существует специальная функция `len()`, которая позволяет измерить длину строки. Результатом выполнения функции является число, равное количеству символов в строке. Для строк доступны операции конкатенации, т. е. объединения, обозначается знаком «+» и дублирования, обозначается знаком «*» (рис. 16).

```
>>> len ("Сколько символов в строке?")
26
>>> "Какая " + "прекрасная " + "погода"
'Какая прекрасная погода'
>>> "0" * 13
'0000000000000'
```

Рисунок 16 – Операция конкатенации

Так как в строке важна последовательность символов (у каждого символа свое положение), то существует возможность обращения к

определенному символу в строке и извлечения его с помощью оператора индексирования, который обозначается квадратными скобками. Необходимо помнить, что нумерация символов начинается с нуля (рис. 17).

```
>>> "Синтаксис" [1]
'и'
>>> "Операция" [4]
'а'
```

Рисунок 17 – Оператор индексирования

При необходимости извлечения символов, находящихся в конце предусмотрена обратная нумерация. Так, чтобы извлечь последний символ не нужно считать все символы в строке, можно использовать нумерацию с отрицательными числами (рис. 18).

```
>>> "Гиперболический параболоид" [-1]
'д'
>>> "Млечный путь" [-4]
'п'
```

Рисунок 18 – Обратная нумерация

В данном случае нумерация с конца начинается с единицы.

В большинстве случаев легче работать не с самими строками, а с переменными, которые ссылаются на них. Результат индексирования можно присвоить другой переменной.

В языке Python есть возможность извлечения сразу нескольких символов из строки. Оператор извлечения выглядит так: [X:Y], где X – начальный индекс, а Y – конечный. При отсутствии начального индекса символы извлекаются начиная с нулевого и до конечного. В случае отсутствия последнего, символы извлекаются с введенного начального индекса и до последнего (рис. 19).

```
>>> a = "Номер пользователя"
>>> a[0:5]
'Номер'
>>> Строка = "На улице снова дождь"
>>> Строка[9:]
' снова дождь'
```

Рисунок 19 – Пример извлечения символов

Помимо этого, можно извлекать символы не подряд, а через определенное количество. В таком случае оператор индексирования будет выглядеть следующим образом: [X:Y:Z], где Z – шаг, через который осуществляется извлечение символов (рис. 20).

```
>>> Строка = "Буква, Забор, Ветка"
>>> Строка[::7]
'БЗВ'
>>> Строка[0:6:2]
'Бка'
```

Рисунок 20 – Оператор извлечения символов

Занятие 9. Последовательности: списки

Тип урока: урок усвоения новых знаний.

Цель урока: познакомить учащихся с основными понятиями «списки».

Задачи урока:

Образовательная: познакомить учащихся с основными понятиями, научить применять на практике.

Развивающая: развивать у учащихся логическое и алгоритмическое мышление, навыки мыслительной деятельности, включая каждого учащегося в учебно-познавательный процесс и создавая условия для работы каждого в индивидуальном психологическом ритме.

Воспитательная: воспитывать у учащихся самостоятельность, активность, интерес к предмету.

Как и строки, списки в языке программирования Python являются упорядоченными последовательностями однотипных данных. Только в отличие от строк, списки состоят из различных объектов (значений, данных), которые перечисляются через запятую и ограничены с двух сторон квадратными скобками. Например: [5,63,-25,735] и [«Москва», «Лондон», «Нью-Йорк»]. Также списки могут состоять из списков: [[1, 0, 0], [0, 1, 0], [0, 0, 1]].

Как и строки списки можно соединять и повторять. И аналогично строкам можно получать доступ к отдельным объектам из списка по их индексам и извлекать несколько объектов (рис. 21).

```
>>> [0,1,2,3,4,5] + [6,7,8,9]
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> [[0,0],[1,0]]*2
[[0, 0], [1, 0], [0, 0], [1, 0]]
>>> X = [1,2,3,4,5]
>>> X[0]
1
>>> X[-1]
5
>>> X[0:2]
[1, 2]
```

Рисунок 21 – Пример извлечения нескольких объектов

Но в отличие от строк, списки можно изменять. Например, заменять один объект из списка на новый или же группы объектов на один и более (рис. 22).

```
>>> Список = ["Молоко", "Черный Хлеб", "Сыр"]
>>> Список[1] = "Белый Хлеб"
>>> Список
['Молоко', 'Белый Хлеб', 'Сыр']
>>> Список[2:4] = ["Колбаса", "Сосиски"]
>>> Список
['Молоко', 'Белый Хлеб', 'Колбаса', 'Сосиски']
```

Рисунок 22 – Пример замены одного объекта другим

На данном занятии следует дать типовые упражнения на закрепление этой и предыдущей тем.

Занятие 10. Решение задач

Задание 1.

Запросить у пользователя ввод числа от 1 до 9 и вывести на экран квадрат этого числа.

Решение (рис. 23):

```
>>> x = int(input("Введите число от 1 до 9: "))
Введите число от 1 до 9: 5
>>> o = x**2
>>> print("Квадрат числа ", x, " равен: ", o)
Квадрат числа 5 равен: 25
```

Рисунок 23 – Результат выполнения

Необходимо пояснить, что возможен вывод и текста, и значения переменной в одном выражении.

Задание 2.

Запросить у пользователя ввод числа a и b , которые должны находиться в промежутке $0 \leq a \leq 10$, и вывести на экран значение числа x , равное возведению числа a в степень числа b .

Решение (рис. 24):

```
>>> a = int(input("Введите число a промежутке от 0 до 10: "))
Введите число a промежутке от 0 до 10: 3
>>> b = int(input("Введите число b промежутке от 0 до 10: "))
Введите число b промежутке от 0 до 10: 3
>>> if a and b >= 0 and a and b <= 10:
    x = a**b
    print("Число", a, "возведенное в степень", b, "=", x)
else:
    print ("Число не удовлетворяет условию")
Число 3 возведенное в степень 3 = 27
```

Рисунок 24 – Результат выполнения

Необходимо пояснить, что функция `if` и включенная в нее функция `else` должны находиться на одном уровне.

Задание 3.

Запросить у пользователя ввод двух слов, присвоить им переменные и найти произведение количества букв первого слова на количество букв второго.

Решение (рис. 25):

```
>>> a = input("Введите первое слово: ")
Введите первое слово: Моя
>>> b = input("Введите второе слово: ")
Введите второе слово: Мечта
>>> la=len(a)
>>> lb=len(b)
>>> x=la*lb
>>> print("Количество символов в первом слове: ", la, ". Количество символов во в
втором слове: ", lb, "Произведение: ", x)
Количество символов в первом слове: 3 . Количество символов во втором слове: 5
Произведение: 15
```

Рисунок 25 – Результат выполнения

Не обязательно выводить последнее сообщение, можно просто сразу вывести произведение.

Задание 4.

Дан список чисел от 1 до 10. Разделить список на два других, первый будет включать первые 5 значений, а второй последние 5 значений. В первом списке к каждому числу прибавить число 5, во втором списке каждое число возвести в степень 2. Вывести отдельно два списка на экран.

Чтобы избежать ошибок и полноценно написать код, то легче использовать режим написания отдельных модулей, после чего ответ будет выведен в среде разработки.

Решение (рис. 26):

```

Список = [1,2,3,4,5,6,7,8,9,10]
Список1 = Список[0:5]
Список2 = Список[5:]
i = 0
for Элемент1 in Список1:
    Список1[i] = Элемент1 + 5
    i = i+1
ii=0
for Элемент2 in Список2:
    Список2[ii] = Элемент2 * 5
    ii = ii+1
print("Первый список: ", Список1, "Второй список: ", Список2)

```

Рисунок 26 – Результат выполнения

Ответ (рис. 27):

```

Первый список: [6, 7, 8, 9, 10] Второй список: [30, 35, 40, 45, 50]
>>>

```

Рисунок 27 – Ответ

На сайте мы представили конспекты разработанного нами курса в разделе «Учебные материалы» (рис. 28).

КОНСПЕКТЫ УРОКОВ

- [Урок 1. Введение в язык программирования Python](#)
- [Урок 2. Решение вычислительных задач](#)
- [Урок 3. Типы данных. Определение переменной](#)
- [Урок 4. Логические выражения](#)
- [Урок 5. Условный оператор. Инструкция if](#)
- [Урок 6. Цикл While](#)
- [Урок 8. Ввод данных с клавиатуры](#)
- [Урок 9. Последовательности: строки](#)
- [Урок 10. Последовательности: списки](#)
- [Урок 11. Решение задач](#)

Создатель сайта Смирнов А. С.

Рисунок 28 – Конспекты уроков, размещенных на сайте

Далее представим блок самостоятельной работы, что было особо актуально в ситуации пандемии 2020 года.

Самостоятельная работа учащихся не должна ограничиваться повторением уже изученного материала и решением типовых заданий, она должна давать каждому учащемуся возможность развить свои знания в данной области, совершенствовать умения, искать творческий подход к решению нестандартных заданий. Исходя из этого определим необходимое количество уровней сложности. Использовать один уровень сложности нецелесообразно: ученики в классе имеют разный уровень подготовки и способностей, при развивающем воздействии подобранных заданий на слабых учениках, для сильных учеников данный уровень сложности будет слишком легким и наоборот: при подходящем для сильных учеников уровне сложности, слабые не будут справляться с заданиями, и их мотивация к дальнейшему изучению программирования будет стремительно ослабевать. При использовании двух уровней сложности происходит большая дифференциация учащихся. Мы уже имеем возможность подобрать задания для слабых учеников отдельно, а для сильных выбрать более сложные задания. Однако, мы советуем использовать три уровня сложности: первый уровень для учащихся, не успевающих на уроке; второй – для основной массы учащихся, в целом справляющихся с заданиями во время занятий; третий – задачи повышенной сложности для тех, кому недостаточно знаний, полученных на уроке. В дальнейшем мы будем использовать три уровня сложности заданий. Определить количество уровней сложности недостаточно. Необходимо так же разработать критерии, согласно которым мы будем относить задание к той или иной категории.

Критерии выбираются исходя из целей, которые ставит учитель перед, результатов, которые ожидаются после выполнения учащимися самостоятельной работы. В нашем случае, мы поставили перед собой целью самостоятельной работы решение учащимися задач, требующих от них

ранее неиспользованного подхода к решению или поиска дополнительной информации.

Несмотря на то, что задачи первого уровня сложности не предусматривают особого поиска информации или нового способа решения, для не успевающих учеников задачи данного уровня в соотношении со сложностью других уровней и учащихся, для которых они предназначены, будут относительно равноценны.

При организации самостоятельной работы так же следует учитывать мотивирующую силу отметки и четко определить для учащихся критерии, по которым будет выставляться отметка. Наиболее объективным в данном случае является определить соотношение набранных баллов и отметки, которая будет выставляться за данную работу. При определении количества баллов и оценки за них, необходимо учитывать количество заданий на каждом из уровней и их сложность (количество баллов за каждое задание).

Основываясь на задачах, решаемых на уроках в классе и предъявленных критериях, подбираем задачи, которые будут использоваться в дальнейшем в качестве заданий для самостоятельной работы. Данные задачи можно как составить, используя различные источники, так и придумать самостоятельно.

Количество заданий каждого уровня сложности для каждого раздела рекомендуется сделать одинаковым, при этом отметим, что в зависимости от уровня сформированности знаний и умений обучающихся уровни могут комбинироваться.

Для нашего курса мы отобрали следующие задачи:

Линейные программы

Уровень 1

1. Дано расстояние в сантиметрах. Найти число полных метров в нем.
2. Дано трехзначное число. Вывести сначала последнюю цифру (единицы). Затем первую цифру (сотни).

3. Известна стоимость монитора, системного блока, клавиатуры и мыши. Сколько будут стоить N компьютеров из этих элементов?

Уровень 2

1. Найдите среднее геометрическое двух чисел (квадратный корень произведения). [8]

2. Даны координаты трех точек A, B, C. Найти сумму всех отрезков (AB, BC, AC).

Уровень 3

Напишите программу, которая на вход будет получать два значения: строку и ключ, а на выходе возвращать строку, зашифрованную путем применения функции XOR (^) над символами строки с ключом.

Условный оператор

Уровень 1

1. Пользователь вводит месяц. Вывести к какому времени года относится.

2. Определить является ли введенный год високосным.

3. Написать программу, принимающую 3 аргумента: первые 2 – числа, третий – операция, которая должна быть произведена над ними. Если третий аргумент «+», сложить их; если «-», то вычесть; «*» – умножить; «:» – разделить (первое на второе). В остальных случаях вернуть строку.

«Неизвестная операция».

Уровень 2

1. Дано целое число. Вывести количество учеников в письменном виде (двенадцать учеников).

2. В восточном календаре принят 60-летний цикл, состоящий из 12-летних подциклов, обозначаемых названиями цвета: синий, красный, желтый, белый и черный. В каждом подцикле года носят названия животных: крысы, коровы, тигра, зайца, дракона, змеи, лошади, овцы, обезьяны, курицы, собаки и свиньи. По номеру года вывести его название, если 1984 был началом цикла – годом зеленой крысы.

Уровень 3

Напишите программу, которая по введенным коэффициентам будет решать квадратное уравнение. Не забудьте предупредить пользователя, если происходит деление на 0 или уравнение не имеет корней.

Циклы

Уровень 1

1. Проверить, является ли введенное число числом Фибоначи.
2. Колхозница привезла на рынок корзину яиц для продажи.

Продавала она их по одной и той же цене. После продажи яиц колхозница пожелала проверить, верно ли она получала деньги. Но вот беда: она забыла, сколько яиц у нее было. Вспомнила она только, что когда перекладывала яйца по 2, то осталось одно яйцо; одно яйцо оставалось так же при перекладывании яиц по 3, по 4, по 5 и по 6. Когда же она перекладывала яйца по 7, то не оставалось ни одного. Помогите колхознице сообразить, сколько у нее было яиц.

3. Вывести квадраты всех чисел от 1 до n .

Уровень 2

1. Пользователь делает вклад в размере a рублей сроком на $years$ лет под 10% годовых (каждый год размер его вклада увеличивается на 10%. Эти деньги прибавляются к сумме вклада, и на них в следующем году тоже будут проценты). Написать программу, по аргументам a и $years$ возвращающую сумму, которая будет на счету пользователя.

2. Дано целое число p . Вывести все его цифры по одной в строке начиная с последней (самой правой).

Уровень 3

Данные целые положительный числа A и B . найти наибольший общий делитель, используя алгоритм Евклида.

Базовые алгоритмы (накопления/поиска максимума(минимума))

Уровень 1

1. Найти сумму и произведение цифр, введенного натурального числа.
2. Дано натуральное число. Вывести на экран все простые числа до заданного включительно. [12]
3. Дано натуральное число n (которое также может быть равно нулю). Вычислить $n!$

Примечание: $n!$ (факториал числа n , читается «эн факториал») – произведение всех натуральных чисел до n включительно.

Уровень 2

1. Дано натуральное число n . Вывести на экран n первых простых чисел.
2. Дано натуральное число n . Проверить, представляет ли собой палиндром его десятичная запись.

Уровень 3

Дано натуральное число. Проверить, является ли оно совершенным.

Примечание: совершенным числом называется натуральное число, равное сумме всех своих собственных делителей (то есть натуральных делителей, отличных от самого числа). Например, 6 – совершенное число, оно имеет три собственных делителя: 1, 2, 3, и их сумма равна $1+2+3 = 6$.

Списки

Уровень 1

1. На вход программа получает количество элементов списка и сами элементы по порядку. После этого еще одно число. Определить, содержит ли список данное число x . Если содержит, то вывести на экран число 7145, если не содержит, то вывести на экран число 5741.
2. На вход программа получает количество элементов списка и сами элементы по порядку. Найти наибольший элемент списка и вывести его на экран.

3. На вход программа получает количество элементов списка и сами элементы по порядку. Поменять местами самый большой и самый маленький элементы списка.

Уровень 2

1. Напишите программу, на вход которой подаётся список чисел одной строкой. Программа должна для каждого элемента этого списка вывести сумму двух его соседей. Для элементов списка, являющихся крайними, одним из соседей считается элемент, находящийся на противоположном конце этого списка. Например, если на вход подаётся список «1 3 5 6 10», то на выход ожидается список «13 6 9 15 7». Если на вход пришло только одно число, надо вывести его же. Вывод должен содержать одну строку с числами нового списка, разделёнными пробелом.

1. Из списка чисел удалить элементы, значения которых больше 35 и меньше 65. При этом удаляемые числа сохранить в другом списке.

Уровень 3

Дан список-массив, заполненный случайным образом нулями и единицами. Найти самую длинную непрерывную последовательность единиц и определить индексы первого и последнего элементов в ней.

Вспомогательные алгоритмы, рекурсия

Уровень 1

1. Напишите функцию, которая будет генерировать список заданного размера и заполнять его случайными числами (`make_list()`), а также функцию, которая будет выводить заданный список (`print_list()`).

2. Напишите программу, которая на вход будет получать количество координат и координаты точек (x, y) и выводить сумму всех отрезков.

3. Напишите функцию, которая определяет количество разрядов введенного целого числа.

Уровень 2

1. В теории вычислимости важную роль играет функция Аккермана $A(m,n)$, определенная следующим образом:

$$\left\{ \begin{array}{l} n + 1, \\ A(m, n) = A(m - 1, 1), \\ A(m - 1, A(m, n - 1)), \\ m = 0; m > 0, n > 0; m > 0, n = 0. \end{array} \right.$$

Даны два целых неотрицательных числа m и n . Выведите $A(m, n)$.

2. Пользователь вводит число. Сообщить, есть ли оно в массиве, элементы которого расположены по возрастанию значений, а также, если есть, в каком месте находится. При решении задачи использовать бинарный (двоичный) поиск, который оформить в виде отдельной функции.

Уровень 3

Написать рекурсивную процедуру, переводящую числа из одной системы счисления в другую.

Примеры конспектов представлены в приложении.

2.3 Апробация результатов исследования в школе

Педагогическая апробация проводилась в рамках научно-исследовательской практики в Муниципальном казенном общеобразовательном учреждении «Основная общеобразовательная школа № 19» г. Коркино Челябинской области. Курс изучался у учеников 8- 9 класса. В течение 10 занятий была рассмотрена тема программирования на языке Python, а также блок задач для самостоятельной работы.

Апробация прошла успешно. Способствовала этому правильная мотивация, цели и задачи изучения темы. Тема курса оказалась знакома для учащихся, и они быстро включились в работу. В таком возрасте дети особенно заинтересованы в получении знаний, особенно если информация наглядная и простая.

Вывод: благодаря разработанному элективному курсу, можно повысить мотивацию учащихся средней школы к изучению раздела

«Программирование и основы алгоритмизации», и к предмету «Информатика» в целом.

Выводы по главе 2

В процессе описания элективного курса были учтены все аспекты теоретической части работы.

В ходе практической части, был проведен анализ нормативных документов. Рассмотрен современный язык программирования Python как наиболее удобный и подходящий в образовательных целях. Были выделены основные умения учащихся, необходимые для изучения курса, поставлена цель и описаны задачи данного курса. Составлен курс по выбору Python для учеников, заинтересованных в углубленном изучении программирования. Составлены требования для учащихся, допущенных к этому курсу. Обозначены цели и задачи, а также описаны знания и умения, полученные после прохождения этого курса.

ЗАКЛЮЧЕНИЕ

Модернизация Российской системы образования делает необходимым поиск новых и современных подходов к организации учебного процесса. Программирование – это остро актуальная сфера деятельности, именно поэтому информатика является одним из наиболее популярных для сдачи предметов.

Целью данной выпускной квалификационной работы является изучение теоретических аспектов темы исследования и разработка элективного курса «Основы программирования на Python».

В результате выполнения работы были проанализированы нормативные документы основного общего образования (ФГОС) на наличие требований к изучаемым языкам программирования и авторские учебные программы по информатике и ИКТ.

Был проведен сравнительный анализ современных распространенных языков программирования на степень возможности обучения на них (соответствие выдвинутым критериям). В результате анализа наиболее оптимальным вариантом для обучения программированию детей оказался язык программирования Python.

Далее было изучено содержание обучения программированию в основной школе, которое входит в раздел «Алгоритмизация и программирование», для более грамотного составления элективного курса «Основы программирования на языке Python». В итоге был разработан элективный курс «Основы программирования на языке Python», рассчитанный на учащихся старшего звена основной школы и приведены методические рекомендации по проведению данного курса. Курс

Исходя из вышеизложенного анализа, а также изучения методических документов был разработан элективный курс, который учитывал все возрастные и психолого-педагогические особенности. Курс изучался у учеников 8-9 класса. В течение 10 занятий была рассмотрена тема программирования на языке Python, а также блок задач для самостоятельной

работы. Также нами были предложены задачи для самостоятельной работы, которые были построены на основе метода эвристического задания.

Предлагаемый курс является интересным и одновременно занимательным, отличается высоким уровнем абстрактности на уровне основ программирования и может быть интересным и полезным для всех школьников.

При изучении языка программирования Python учащиеся с легкостью смогут освоить синтаксис, в котором используются минимальное количество вспомогательных слов. Уже через несколько занятий смогут писать отдельные модули программ для дальнейшего внедрения и будут развивать способности, без которых в будущем сложно будет представить современное общество. В свою очередь учащиеся, которые хотят далее связать свою профессию с информационными технологиями смогут успешно подготовиться к экзаменам, так как все задачи, связанные с программированием, включают также описания на языке Python.

Данная разработка может быть полезна как начинающим учителям информатики, так и опытным учителям, решившим начать преподавание нового языка программирования.

Таким образом, цель работы достигнута.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Абрамовских, В. В. Язык программирования Python в качестве дополнительного языка программирования в школе [Текст] / В. В. Абрамовских // XIX Всероссийская студенческая научно-практическая конференция Нижневартковского государственного университета сборник статей. – 2017. – С. 16-18.
2. Бизли, Д. М. Язык программирования Python. Справочник [Текст] / Д. М. Бизли, Г. Ван Россум Босова Л. Л. Босова. – Москва : ДиаСофт, 2000. – 858 с. – - ISBN 966-7393-54-2.
3. Бобров, А. Н. Проблемы выбора языка программирования в школьном курсе информатики [Текст] / А. Н. Бобров // Молодой ученый. – 2015. – № 24 (104). – С. 61-64. – ISSN: 2072-0297.
4. Бородин, М. Н. Информатика. Программы для общеобразовательных учреждений. 2-11 классы [Текст] / М. Н. Бородин. – Москва : Бинوم. Лаб. знаний, 2012. – 584 с. – ISBN 978-5-9963-1169-9.
5. Босова, Л. Л. Информатика 7 класс [Текст] / Л. Л. Босова, А. Ю. Босова. – 2-е изд. – Москва : БИНОМ, 2014. – 224 с. – ISSN: 978-5-09-079968-3.
6. Босова, Л. Л. Информатика 9 класс [Текст] / Л. Л. Босова, А. Ю. Босова. – 2-е изд. – Москва : БИНОМ, 2014. – 184 с. – ISBN 978-5-09-079645-3.
7. Босова, Л. Л. Информатика 8 класс [Текст] / Л. Л. Босова, А. Ю. Босова. – 2-е изд. – Москва : БИНОМ, 2014. – 157 с. – ISBN 978-5-09-079645-3.
8. Бриггс, Д. Python для детей. Самоучитель по программированию [Текст] / Д. Бриггс. – Москва : Манн, Иванов и Фербер, 2018. – 320 с. – ISBN 978-5-00100-616-9.
9. Гейн, А. Г. Информатика. Методические рекомендации. 11 класс [Текст] : учеб. пособие для общеобразовательных организаций /

А. Г. Гейн, Н. А. Юнерман, А. А. Гейн. - 2-е изд. – Москва : Просвещение, 2017. – 240 с. – — ISBN 978-5-09-044797-3.

10. Душкин, А. В. Роль элективных курсов по информатике в профессионально-образовательной траектории обучающихся [Текст] / А. В. Душкин // Вестник науки и образования. – 2016. – № 11 (23) – С. 100-102. – ISSN: 2312-8089.

11. Есауленко, В. Г. Язык Python как основной язык программирования в школе [Текст] / В. Г. Есауленко // Педагогическое образование на Алтае. – №1. – 2017. – С. 48-50. – ISSN: 2413-449X.

12. Информатика и ИКТ [Текст] : учебник для 10-11 кл. общеобразоват. учреждений: базовый и профил. уровни / А. Г. Гейн, А. Б. Ливчак, А. И. Сенокосов, Н. А. Юнерман. – Москва : Просвещение, 2012. – 256 с. – ISBN 978-5-09-019446-4.

13. Информатика. 10-11 классы. Базовый уровень [Текст]: методическое пособие / Н. В. Макарова, Ю. Ф. Титова, Ю. Н. Нилова и др.; под ред. Н. В. Макаровой. – Москва : БИНОМ. Лаборатория знаний, 2016. – 336 с. – ISBN 978-5-9963-3131-4.

14. Кириенко, Д. П. Язык программирования Python – современный язык для обучения [Текст] / Д. П. Кириенко // Всероссийский съезд учителей информатики. – Москва : МГУ имени М.В. Ломоносова, 2011. – С. 358-359.

15. Кочеткова, О. А. Разработка электронных средств учебного назначения по курсу «Программирование на языке Python» [Текст] / О. А. Кочеткова, И. В. Долгополов // Университетское образование (МКУО-2016): сборник статей XX Международной научно-методической конференции (Пенза, 7-8 апреля 2016 г.). – Пенза: Издательство Пензенского государственного университета – 2016. – С. 104-105.

16. Кочеткова, О. А., Обучение учащихся программированию на языке python в рамках элективного курса по информатике [Электронный ресурс] / О. А. Кочеткова, Ю. Н. Пудовкина // Современные проблемы

науки и образования. – 2019. – № 2. URL: <http://science-education.ru/ru/article/view?id=28708>, свободный. – Загл. с экрана. – Яз. рус.

17. Лапчик, М. П. Методика преподавания информатики [Текст] / М. П. Лапчик, И. Г. Семакин, Е. К. Хенер. – Москва : Академия, 2017. – 622 с. – ISBN 978-5-7695-4502-3.

18. Лапшева, Е. Е. Введение языка программирования Python в школьный курс информатики [Текст] / Е. Е. Лапшева // Компьютерные науки и информационные технологии: Материалы Междунар. науч. конф. – Саратов: Издат. центр «Наука», 2016. – С. 232-233.

19. Левченко, И. В. Методологические вопросы методики обучения информатике в средней общеобразовательной школе [Текст] / И. В. Левченко. – Москва : МГПУ, 2012. – 123 с.

20. Левченко, И. В. Общие вопросы методики обучения информатике в средней школе [Текст] : учебное пособие для студентов педагогических вузов и университетов / И. В. Левченко. – Москва : МГПУ, 2003. – 105 с.

21. Лутц, М. Python : Карманный справочник [Текст] / М. Лутц. – Москва : Вильямс, 2016. – 320 с. – ISBN 978-5-8459-1965-6.

22. Малеев, В. В. Общая методика преподавания информатики [Текст]: учеб. пособие / В. В. Малеев. – Воронеж: ВГПУ, 2005. – 271 с. – ISBN 5-88519-276-6.

23. Наумов, Р. В. Актуальные языки программирования [Текст] / Р. В. Наумов // Academy. – 2016. – №3. – С. 54-56.

24. Окулов, С. М. Основы программирования [Текст] / С. М. Окулова – Москва : Бином, 2012. – 336 с. – ISBN 978-5-9963-1932-9.

25. Поляков, К. Ю. Информатика. Базовый и углубленный уровни. (в двух частях), 10 класс [Текст] / К. Ю. Поляков, Е. А. Еремин. – Москва : «Бином. Лаборатория знаний», 2013. – 344 с. – ISBN 978-5-9963-3136-9.

26. Поляков, К. Ю. Информатика. 10-11 классы. Базовый и углубленный уровни [Текст] : методическое пособие / К. Ю. Поляков, Е. А. Еремин. – Москва : БИНОМ. Лаб. знаний, 2016. – 128 с. – ISBN 978-5-9963-1568-0.
27. Поляков, К. Ю. Информатика. 7 класс / К. Ю. Поляков, Е. А. Еремин. – Москва : БИНОМ, 2017. – 302 с. – ISBN 978-5-9963-4585-4.
28. Поляков, К. Ю. Информатика. 8 класс / К. Ю. Поляков, Е. А. Еремин. – Москва : БИНОМ, 2018. – 253 с. – ISBN 978-5-9963-3749-1.
29. Поляков, К. Ю. Информатика. 9 класс [Текст] / К. Ю. Поляков, Е. А. Еремин. – Москва : БИНОМ, 2017. – 276 с. – ISBN 978-5-9963-3109-3.
30. Поляков, К. Ю. Информатика. Базовый и углубленный уровни. (в двух частях) 11 класс [Текст] / К. Ю. Поляков, Е. А. Еремин. – Москва : «Бином. Лаборатория знаний», 2013. – 304 с. – ISBN 978-5-9963-3139-0.
31. Поляков, К. Ю. Язык Python глазами учителя [Текст] / К. Ю. Поляков // Информатика. – 2014. – № 9. – С. 4-16.
32. Программирование на языке Python для школьников: Учебное пособие по изучению языка программирования Python [Текст] / Л. Самыкбаева, А. Беляев, А. Палитаев, И. Ташиев, С. Маматов – Фонд Сорос-Кыргызстан, 2019 – 84 с. – ISBN 978-9967-31-911-0.
33. Прохоренок, Н. А. Python. Самое необходимое [Текст] / Н. А. Прохоренок. – Санкт-Петербург : БХВ-Петербург, 2011. – 416 с. – ISBN 978-5-9775-0614-4.
34. Родыгин, Е. Ф. Методические рекомендации обучения программированию в школе [Электронный ресурс] / Е. Ф. Родыгин // Вестник Марийского государственного университета. – 2015. – Режим доступа : <https://cyberleninka.ru/article/v/metodicheskie-rekomendatsii-obucheniya-programirovaniyu-v-shkole>, свободный. – Загл. с экрана. – Яз. рус.

35. Саммерфилд, М. Программирование на Python 3. Подробное руководство (пер. с англ.) [Текст] / М. Саммерфилд. – Санкт-Петербург : Символ-Плюс, 2009. – 608 с. – ISBN 978-5-93286-161-5.
36. Семакин, И. Г. Информатика. 7 класс [Текст] / И. Г. Семакин, Л. А. Залогова. – Москва : БИНОМ, 2015. – 167 с. – ISBN 978-5-9963-1964-0.
37. Семакин, И. Г. Информатика. 8 класс [Текст] / И. Г. Семакин, Л. А. Залогова. - 3-е изд. – Москва : БИНОМ, 2015. – 174 с. – ISBN 978-5-9963-0274-1.
38. Семакин, И. Г. Информатика. 9 класс [Текст] / И. Г. Семакин, Л. А. Залогова. - 3-е изд. – Москва : БИНОМ, 2015. – 207 с. – ISBN 978-5-9963-3522-0.
39. Сорокина, Н. А. Python как основной язык программирования в средней школе [Текст] / Н. А. Сорокина // Молодой ученый. – 2019. – № 5 (243). – С. 15-16.
40. Сузи Р.А. Python. Наиболее полное руководство [Текст] / Р. А. Сузи – Санкт-Петербург : БХВ-Петербург, 2002. – 759 с. – ISBN 5-94157-097-X.
41. Сузи, Р. А. Язык программирования Python [Текст] / Р. А. Сузи – Москва : Бином. Лаборатория знаний, 2006. – 328 с. – ISBN 5-9556-0058-2.
42. Угринович, Н. Д. Преподавание курса «Информатика и ИКТ» в основной и старшей школе [Текст] : метод. пособие / Н. Д. Угринович. – Москва : БИНОМ. Лаборатория знаний, 2006. – 182 с. – ISBN 978-5-94774-589-4.
43. Федеральный Государственный образовательный стандарт (основного) общего образования [Электронный ресурс] : утв. приказом Минобрнауки России от 17.12.2010 № 1897 / Рос. Федерация. Минобрнауки России. – Режим доступа: <https://base.garant.ru/55170507/53f89421bbdaf741eb2d1ecc4ddb4c33/>, свободный. – Загл. с экрана. – Яз. рус.

44. Федорова Н.Е. Структура, содержание и методические подходы к преподаванию языков программирования в школе [Текст] / Н. Е. Федорова // Современные информационные технологии и ИТ-образование. – 2011. – № 7. – С. 892-897.

45. Фридланд, А. Я. Информатика и компьютерные технологии. Основные термины. Толковый словарь. [Текст] / А. Я. Фридланд, Л. С. Ханамирова, И. А. Фридланд. – Санкт-Петербург : Астрель, 2016. – 272 с. – ISBN 5-88705-032-2.

46. Хахаев, И. А. Практикум по алгоритмизации и программированию на Python. [Текст] / И. А. Хахаев – Москва :Альт Линукс, 2015. – 126 с. – ISBN 978-5-905167-02-7.

47. Чаплыгин, А. Н. Учимся программировать вместе с Питоном [Электронный ресурс] / А. Н. Чаплыгин. – Москва : Бином, 2004. – 110 с. – Режим доступа: <http://gogotor.online/chapligin-a-n-uchimsya-programirovat-vmeste-s-pitonom-python-t173126.html>, свободный. – Загл. с экрана. – Яз. рус.

ПРИЛОЖЕНИЕ

Конспект урока на тему: Введение в язык программирования Python

Тип урока: урок усвоения новых знаний.

Цель урока: познакомить учащихся с особенностями языка программирования Python, изучить основы программирования на данном языке.

Задачи урока:

Образовательная: сформировать представление о языке Python

Развивающая: развивать у учащихся логическое и алгоритмическое мышление, навыки мыслительной деятельности, включая каждого учащегося в учебно–познавательный процесс и создавая условия для работы каждого в индивидуальном психологическом ритме;

Воспитательная: способствовать формированию самооценки (саморефлексии).

Этапы урока:

Организационный момент – 1 мин.

Изучение нового материала – 25 мин.

Рефлексия – 2 мин.

Закрепление изученного материала – 17 мин.

Оборудование: Компьютер, Python (версия не ниже 3.5), модули Tkinter и NumPy, среды разработки на Python: IDLE, Eric или Geany, а также какие-либо эмуляторы терминалов _ xterm, gxvt, проектор.

Структура и ход урока

1. Организационный момент 1 мин.

Учитель: Приветствует класс, проверяет присутствующих.

Здравствуйте, ребята. Сегодня мы с вами познакомимся с языком программирования

Python

2. Изучение нового материала – 25 мин.

Учитель: Демонстрирует презентацию и рассказывает новый материал

История языка программирования.

Основным разработчиком ЯП Python является Гвидо ван Россум (Guido van Rossum), первая стабильная версия 1.0 появилась в январе 1994 года. После того как Гвидо разработал Python, он его выложил в интернет, после этого над его улучшением читало работать целое сообщество программистов. Официальный сайт: <http://python.org>.

Рассмотрим особенности языка.

Python – это интерпретируемый ЯП. Python имеет достаточно простой синтаксис. Код на этом языке программирования достаточно легко читается, т.к. используется в нем минимум вспомогательных элементов, а стиль кода жестко продиктован стандартом PEP-8, в котором четко прописано как писать программы.

Python язык высокого уровня: поддерживает объектно-ориентированное программирование. Высокоуровневый – обозначает, что вы будете писать код при помощи самых обычных слов на английском языке, поэтому код будет легко читаться. Также он считается полноценным и универсальным ЯП. На данном ЯП вы сможете разрабатывать что угодно: веб сайты, игры, программы под компьютер, под телефон, различные скрипты, плагины, моды, и. т.д.

Распространяется Python свободно под лицензией подобной GNU General Public License.

Интерактивный режим.

Как уже было сказано интерпретатор выполняет команды построчно, т.е. пишем строку -> интерпретатор выполняет ее -> наблюдаем результат.

Это очень удобно, когда человек только изучает программирование или тестирует какую-нибудь небольшую часть кода. Ведь если работать на компилируемом языке, то пришлось бы сначала написать код на исходном

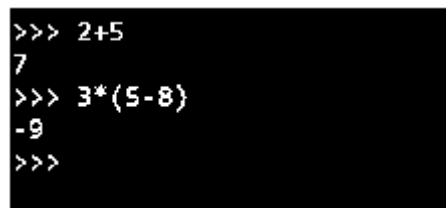
языке программирования, затем скомпилировать и уже потом запустить получившийся файл (с машинным кодом) на исполнение. Если окажется, что где-то в исходном коде была допущена ошибка, то придется перекомпилировать всю программу. Но в интерпретируемых языках нет такой проблемы.

Работать в интерактивном режиме можно в консоли. Для этого следует выполнить команду `Python`. Запустится интерпретатор, где сначала выведется информация о его версии и иная информация. Далее, приглашение к вводу (`>>>`).

Задание. Запустите интерпретатор Питона.

Начнем с простого, поскольку никаких команд мы пока не знаем, то будем использовать

Питон как калькулятор (возможности языка это позволяют).



```
>>> 2+5
7
>>> 3*(5-8)
-9
>>>
```

Рисунок 1– Использование Python в качестве калькулятора

Далее, набирайте подобные примеры в интерактивном режиме (в конце каждого нажимайте `Enter`).

Ответ выдается сразу после нажатия `Enter` (завершения ввода команды). Бывает, что в процессе ввода допустили ошибку или требуется повторить ранее используемую команду.

Чтобы не писать строку сначала, в консоли можно прокручивать список команд, используя для этого стрелки на клавиатуре.

Следующий вариант работы в интерактивном режиме – это работа в среде разработки `IDLE`, у которой есть интерактивный режим работы. `IDLE` нужна для того чтобы вбивать прямо в ней какие-то команды, на языке `Python`. Также `IDLE` может использоваться и как полноценный редактор кода. В отличие от консольного варианта тут мы можем наблюдать

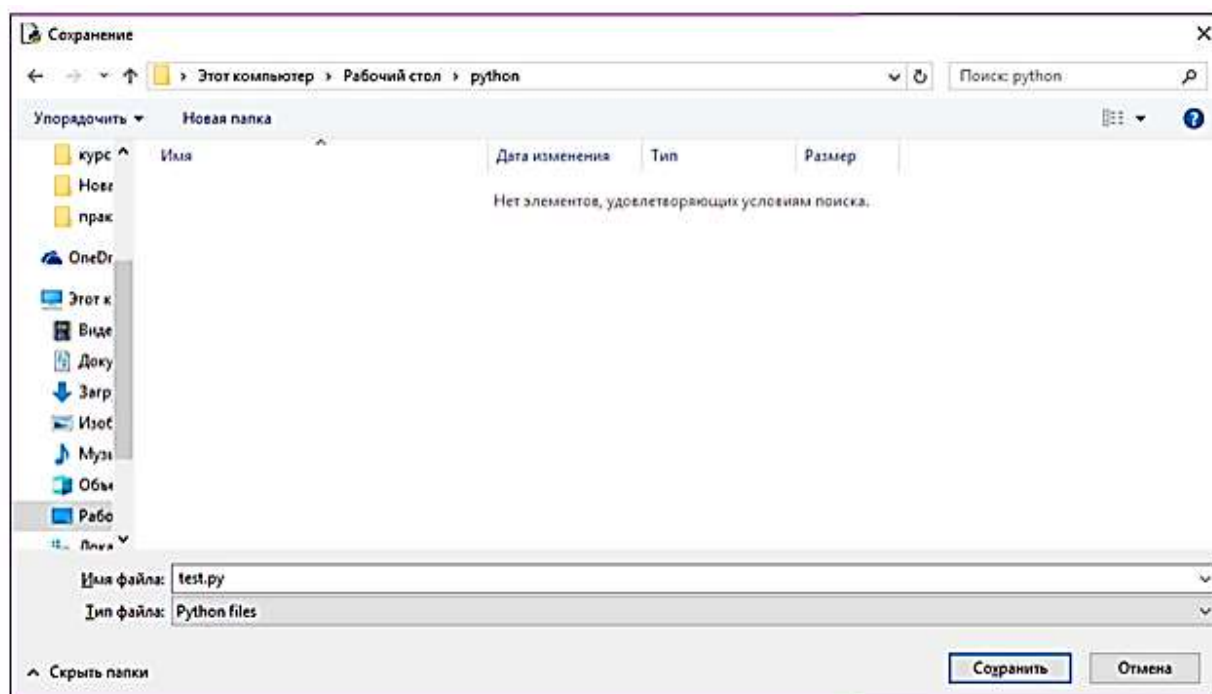
подсветку синтаксиса – все команды выделяются цветом. С помощью комбинаций Alt+N, Alt+P можно прокручивать список команд.

Чтобы запустить IDLE, нажмите на пуск и вбивайте IDLE, запускаем, открывается окошко, это и есть интерактивная оболочка IDLE. Она нужна для того чтобы прямо здесь какие-то команды на языке python. Это используют новички для того чтобы тренироваться, смотреть как работают команды, тестировать модули и так далее. IDLE так же может использоваться и как полноценный редактор кода.

Чтобы открыть редактор кода, нужно нажать file – New File. Открывается окошко, в

котором вы можете вбивать абсолютно любой код, любой длины в любом количестве. После того как вы вписали весь нужный код, нажмите сохранить, и запустите клавишей F5 или Run –

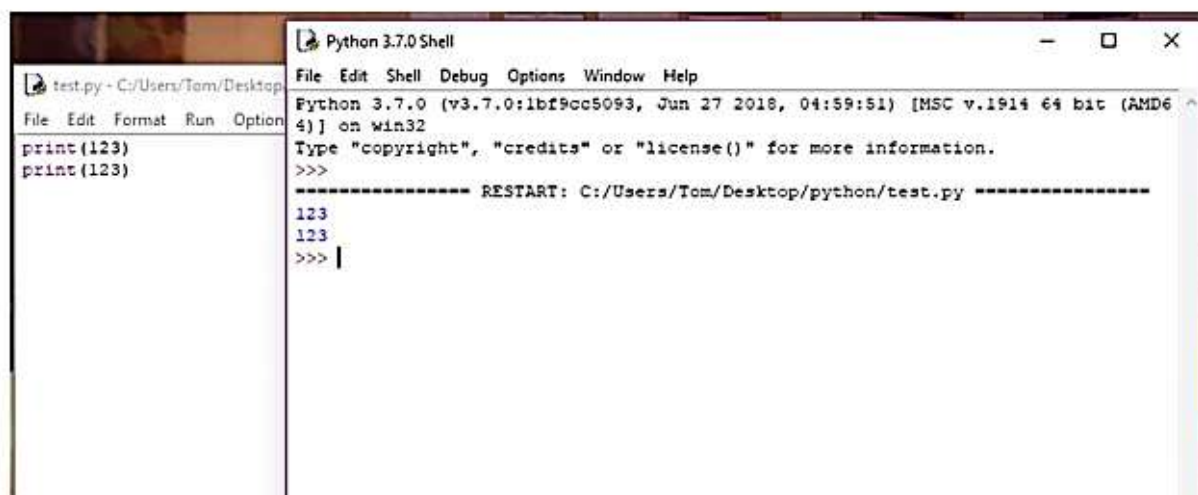
Run Module. Если файл не сохранен, он не сможет запустить и предложит сохранить. Название можно указать любое, обязательно с расширением .py



Например запишем команду

```
print(123)
```

print(123) и нажмем F5, и как видим, IDLE запустил этот файл.



The image shows two overlapping windows. The background window is a text editor titled 'test.py - C:/Users/Tom/Desktop'. It contains the following code:

```
print(123)
print(123)
```


The foreground window is titled 'Python 3.7.0 Shell'. It shows the Python interpreter's output for the code above:

```
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:51) [MSC v.1914 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
----- RESTART: C:/Users/Tom/Desktop/python/test.py -----
123
123
>>> |
```

Код программы пишется в текстовом файле, потом сохраняется с расширением *.py.

Подготовить скрипты можно в той же среде IDLE. Чтобы открыть редактор кода, нужно нажать File > New File. Открывается окошко, в котором вы можете писать абсолютно любой код, любой длины, в любом количестве. Затем желательно сразу сохранить файл в расширении *.py. Если набирать код, не сохранив файл в начале, то синтаксис не будет подсвечиваться.

После того как вписали весь нужный код, сохраните файл ещё раз, чтобы обновить сохранение.

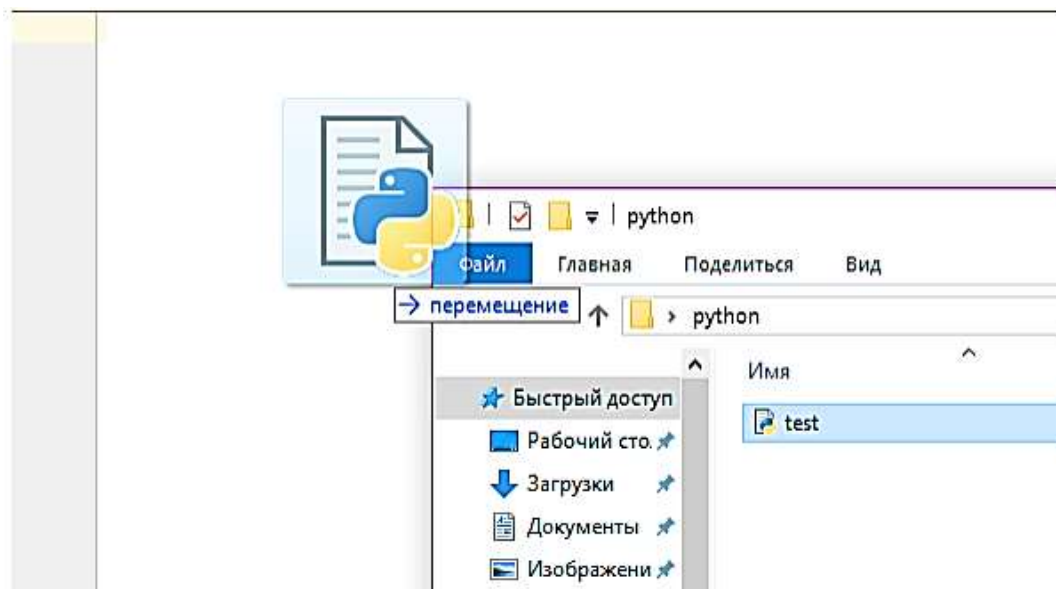
Теперь можно запустить скрипт, выполнив команду меню Run > Run Module или нажать клавишу F5. Если вы не сохранили, то всплывающее окошко предложит вам сначала сохраниться. После этого в первом окне (где «работает» интерпретатор) IDLE автоматически запускает файл. Там где мы писали код, можно нажать F5 и файл снова запустится.

Скрипты можно также писать в любом текстовом редакторе. Также существуют специальные программы для разработки, которые предоставляют дополнительные возможности и удобства.

Запускать подготовленные файлы можно не только в IDLE, но и в консоли с помощью команды python адрес/имя_файла.

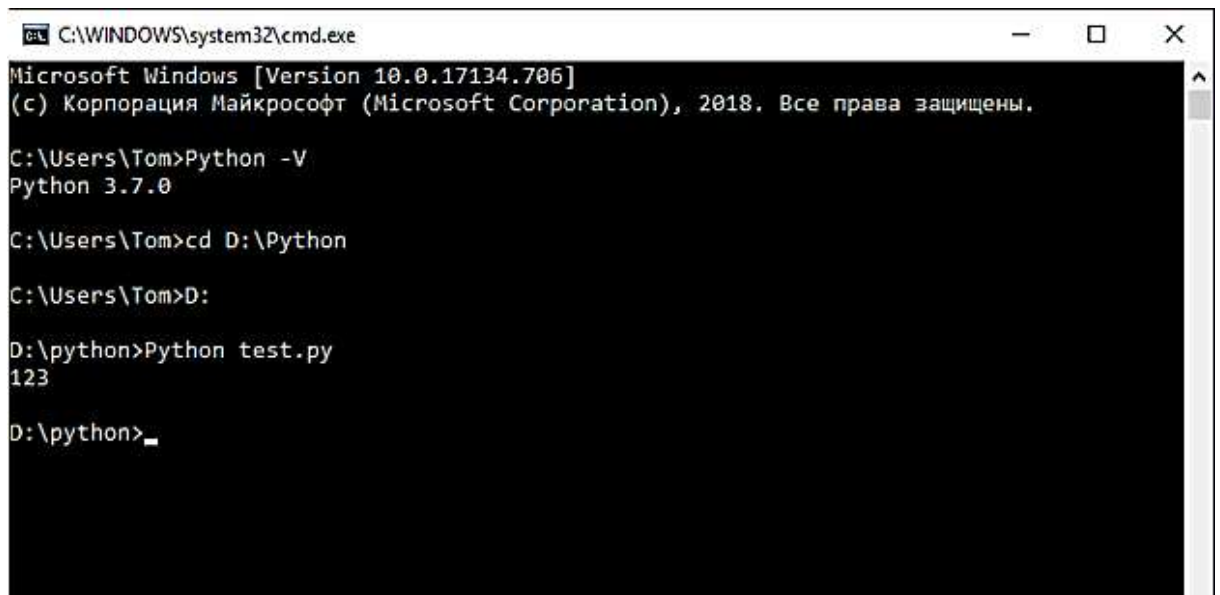
Также, существует возможность настроить выполнение скриптов с помощью двойного клика по файлу (в Windows данная возможность

присутствует изначально). Можно использовать какой-нибудь внешний редактор, и запускать код через консоль. Редакторов для написания кода на python может быть много, тут каждый может использовать то, что удобнее. Можно использовать любой текстовый редактор.



Для того чтобы открыть файл во внешнем редакторе, его нужно просто перетянуть. Для того чтобы запустить код написанный во внешнем редакторе кода, нам понадобится снова командная строка. Из командной строки нужно перейти в ту директорию в которой находится наш скрипт.

1. Мы сохранили файл на диске D в папке Python, поэтому вбиваем команду `cd D:\Python` нажимаем ввод
2. Затем меняем букву диска на D, в следующей строке вводим `D:` нажимаем ввод.
3. В следующей строке вводим команду `Python test.py`, где «test.py» - название нашего файла, нажимаете ввод.



```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.17134.706]
(c) Корпорация Майкрософт (Microsoft Corporation), 2018. Все права защищены.

C:\Users\Tom>Python -V
Python 3.7.0

C:\Users\Tom>cd D:\Python

C:\Users\Tom>D:

D:\python>Python test.py
123

D:\python>_
```

Теперь мы видим что наш код выполнен.

Итак, мы познакомились с двумя вариантами как начать писать и как запускать код python.

Закрепление изученного материала – 17 мин.

Учитель задает вопросы:

- 1) Назовите эти 2 варианта
- 2) Как сохранить код в файл?

Конспект урока на тему: Решение вычислительных задач

Тип урока: урок усвоения новых знаний.

Цель урока: научить учащихся самостоятельно решать вычислительные задачи.

Задачи урока:

Образовательная: научить решать элементарные задачи в Python

Развивающая: развивать у учащихся логическое и алгоритмическое мышление, навыки мыслительной деятельности, включая каждого учащегося в учебно-познавательный процесс и создавая условия для работы каждого в индивидуальном психологическом ритме;

Воспитательная: воспитывать у учащихся самостоятельность, активность, интерес к предмету.

Этапы урока:

Организационный момент – 1 мин.

Повторение пройденного материала – 10 мин.

Рефлексия – 2 мин.

Самостоятельная работа – 22 мин.

Оборудование: Компьютер, Python (версия не ниже 3.5), модули Tkinter и NumPy, среды разработки на Python: IDLE, Eric или Geany, а также какие-либо эмуляторы терминалов _ xterm, gxvt, проектор.

Структура и ход урока

1. Организационный момент 1 мин.

Учитель: Приветствует класс, проверяет присутствующих.

Здравствуйте, ребята. Сегодня мы с вами будем учиться самостоятельно решать вычислительные задачи, и применять полученные знания на практике.

Давайте повторим пройденный материал на прошлом уроке

1. Какие типы данных вы знаете?
2. Как осуществлять ввод и вывод данных?
3. Как задать переменную?

Дальше ученики отвечают, и учитель выдает задания для выполнения на компьютере.

Задание 1. Определить объем цилиндра.

Пример выполнения задания:

```
r=input('введите радиус')
```

```
h= input('введите высоту')
```

```
pi=3,14
```

```
v=pi*r^2*h
```

```
print(v)
```

```
1 r = int(input('введите радиус'))
2 h = int(input('введите высоту'))
3 pi = 3.14
4 v = pi ^ (r ^ 2) ^ h
5 print(v)
```

Рисунок 1 – Нахождение объема цилиндра

Задание 2. Извлеките кубический корень из суммы двух чисел вводимых с клавиатуры.

Пример выполнения задания:

```
x=int(input('введите первое число'))
```

```
y= int(input('введите введите второе число'))
```

```
p=(x+y)**(1/3)
```

```
print(int(p))
```

```
1 x=int(input('введите первое число'))
2 y= int(input('введите введите второе число'))
3 p=(x+y)**(1/3)
4 print(int(p))
5 |
```

Рисунок 2 – Извлечение кубического корня

Ученики выполняют, обращаются за помощью к учителю, при выполнении показывают результат