РАССМОТРЕНО

Руководитель Точки Роста

_____

Абдурахманов Ю.Ж.

Приказ №145 от «31» 08 2023г.

УТВЕРЖДЕНО

Директор

_____

Бартиханов М.М.

Приказ №145 от «31» 08 2023 г.

Общеобразовательная общеразвивающая

программа технической направленности

«Информатика для 8 класса»

Целевая аудитория: обучающиеся 8 класса

Срок реализации: 68 часов

Чиркей, 2023г

УДК
ББК

Программа школьного курса «Информатика» для 8 класса
Авторы:

Целевая аудитория:
Срок реализации:

ISBN

Оглавление

1. Пояснительная записка

Программа данного курса посвящена обучению школьников различным аспектам программирования на современном языке Python. Занятия курса направлены на развитие мышления, логики, творческого потенциала учеников. Программа ориентирована на использование получаемых знаний для разработки реальных проектов. Курс содержит большое количество творческих заданий (именуемых Кейсами).

Цель и задачи обучения

Целью изучения предмета «Информатика» является получение теоретических и практических знаний, умений и навыков в области современной информатики; формирование целостного мировоззрения, соответствующего современному уровню развития науки и общественной практики, учитывающего социальное, культурное, языковое, духовное многообразие современного мира.

Для достижения поставленной цели необходимо решение следующих задач:

- создание условий для развития интеллектуальных и творческих способностей учащихся, необходимых для успешной социализации и самореализации личности;
- формирование информационной и алгоритмической культуры;
- развитие алгоритмического мышления, необходимого для профессиональной деятельности в современном обществе; развитие умений составить и записать алгоритм;
- формирование умений формализации и структурирования информации, умения выбирать способ представления данных в соответствии с поставленной задачей;
- овладение важнейшими общеучебными умениями и универсальными учебными действиями (формулировать цели деятельности, планировать ее, находить и обрабатывать необходимую информацию из различных источников, включая Интернет и др.);

2. Общая характеристика учебного предмета

Программа по предмету «Информатика» предназначена для изучения курса информатики учащимися основной школы. Она включает в себя пять блоков:

- Основы языка Python
- Создание приложений с помощью tkinter
- Криптография
- Искусственный интеллект
- Продвинутые библиотеки языка Python. Pygame

Важная задача изучения этих содержательных линий в курсе – добиться систематических знаний, необходимых для самостоятельного решения задач, в том числе и тех, которые в самом курсе не рассматривались. На протяжении всего курса учащиеся изучают различные аспекты программирования на современном языке Python.

Технологии, используемые в образовательном процессе:

- Технологии традиционного обучения для освоения минимума содержания образования в соответствии с требованиями стандартов; технологии, построенные на основе объяснительно-иллюстративного способа обучения. В основе – информирование, просвещение обучающихся и организация их репродуктивных действий с целью выработки у школьников общеучебных умений и навыков.
- Технологии компьютерных практикумов.
- Технологии реализации межпредметных связей в образовательном процессе.

- Технологии дифференцированного обучения для освоения учебного материала обучающимися, различающимися по уровню обучаемости, повышения познавательного интереса.
- Технология проблемного обучения с целью развития творческих способностей обучающихся, их интеллектуального потенциала, познавательных возможностей. Обучение ориентировано на самостоятельный поиск результата, самостоятельное добывание знаний, творческое, интеллектуально-познавательное усвоение учениками заданного предметного материала.
- Личностно-ориентированные технологии обучения, способ организации обучения, в процессе которого обеспечивается всемерный учет возможностей и способностей обучаемых и создаются необходимые условия для развития их индивидуальных способностей.
- Информационно-коммуникационные технологии.
- Технология коллективных методов обучения (работа в парах постоянного и сменного состава)

Формы организации образовательного процесса: ффронтальные, групповые, индивидуальные, индивидуально-групповые, практикумы; урок-консультация, урок-практическая работа, уроки с групповыми формами работы, уроки-конкурсы.

3. Место курса в учебном плане

Данная программа предусматривает на реализацию программы по информатике в 8 классе 68 часов. Рабочая программа рассчитана на 34 учебные недели, 2 часа в неделю, общее количество часов — 68. Рабочая программа может реализовываться с использованием электронного обучения (ЭО) и дистанционных образовательных технологий (ДОТ).

4. Личностные, метапредметные и предметные результаты освоения учебного предмета

Личностными результатами, формируемыми при изучении предмета информатика, являются:

- формирование ответственного отношения к учению, готовности и способности обучающихся к саморазвитию и самообразованию на основе мотивации к обучению и познанию, осознанному выбору и построению дальнейшей индивидуальной траектории образования на базе ориентировки в мире профессий и профессиональных предпочтений, с учётом устойчивых познавательных интересов;
- формирование целостного мировоззрения, соответствующего современному уровню развития науки и общественной практики, учитывающего социальное, культурное, языковое, духовное многообразие современного мира;
- формирование коммуникативной компетентности в общении и сотрудничестве со сверстниками, детьми старшего и младшего возраста, взрослыми в процессе образовательной, общественно полезной, учебно-исследовательской, творческой и других видов деятельности.

Метапредметные результаты изучения предмета «Информатика»:

- умение самостоятельно определять цели своего обучения, ставить и формулировать для себя новые задачи в учёбе и познавательной деятельности, развивать мотивы и интересы своей познавательной деятельности;
- умение самостоятельно планировать пути достижения целей, в том числе альтернативные, осознанно выбирать наиболее эффективные способы решения учебных и познавательных задач;
- умение соотносить свои действия с планируемыми результатами, осуществлять контроль своей деятельности в процес-

се достижения результата, определять способы действий в рамках предложенных условий и требований, корректировать свои действия в соответствии с изменяющейся ситуацией;
- умение оценивать правильность выполнения учебной задачи, собственные возможности её решения;
- владение основами самоконтроля, самооценки, принятия решений и осуществления осознанного выбора в учебной и познавательной деятельности;
- умение определять понятия, создавать обобщения, устанавливать аналогии, классифицировать, самостоятельно выбирать основания и критерии для классификации, устанавливать причинно-следственные связи, строить логическое рассуждение, умозаключение (индуктивное, дедуктивное и по аналогии) и делать выводы;
- умение создавать, применять и преобразовывать знаки и символы, модели и схемы для решения учебных и познавательных задач;
- умение организовывать учебное сотрудничество и совместную деятельность с учителем и сверстниками; работать индивидуально и в группе: находить общее решение и разрешать конфликты на основе согласования позиций и учёта интересов; формулировать, аргументировать и отстаивать своё мнение;
- формирование и развитие компетентности в области использования информационно-коммуникационных технологий.

Предметные результаты изучения предмета «Информатика»:
- формирование представления об основных изучаемых понятиях курса;
- формирование информационной и алгоритмической культуры; формирование представления о компьютере как универсальном устройстве обработки информации; развитие основных навыков и умений использования компьютерных устройств;

- развитие алгоритмического мышления, необходимого для профессиональной деятельности в современном обществе; развитие умений составить и записать алгоритм для решения конкретной задачи;
- формирование умений формализации и структурирования информации, умения выбирать способ представления данных в соответствии с поставленной задачей, с использованием соответствующих программных средств обработки данных;
- знакомство с базовыми конструкциями языка Python; формирование умения придумывать алгоритмы и их реализовывать на языке Python;
- формирование умений работы с дополнительными библиотеками языка Python (tkinter, pygame, etc);
- формирования представления о том, что такое криптография, каковы были классические алгоритмы шифрования данных в древности и в чем заключаются их недостатки, каковы современные методы шифрования;
- формирование умения создавать реальные приложения с помощью языка Python, формирование умения применять накопленные знания для решения практических задач;
- использование готовых прикладных компьютерных программ по выбранной специализации;
- развитие умений применять изученные понятия, результаты, методы для решения задач практического характера и задач из смежных дисциплин с использованием при необходимости справочных материалов, компьютера;
- формирование навыков и умений безопасного и целесообразного поведения при работе с компьютерными программами и в Интернете, умения соблюдать нормы информационной этики и права.

5. Содержание курса

Основы языка Python (12 часов)
Ввод-вывод данных. Типы данных. Работа со строками. Списки. Условная инструкция. Циклы for и while. Функции. Разработка несложных консольных приложений.

Создание приложений с помощью tkinter (16 часов)
Работа с модулем tkinter. Виджеты. Конфигурация виджетов. Реакция на события. Упаковщики виджетов. Рисование на холсте canvas. Управление нарисованными объектами с помощью клавиатуры. Разработка и создание GUI-приложений “пинг-понг”, “сапер”.

Криптография (26 часов)
История криптографии. Знаменитые шифры (атбаш, сцитала, шифр Цезаря, квадрат Полибия, решетка Кардано). Создание криптографического приложения с помощью tkinter. Шифры, которые практически невозможно разгадать (шифр Виженера). Современные алгоритмы шифрования. Открытый и закрытый ключи. Электронная подпись. Кодирование текста. Работа с файлами в Python. Продвинутые возможности Python: словари. Дополнительные библиотеки языка Python для работы с датами и временем. Разработка игрового приложения “Мемори”.

Искусственный интеллект (4 часа)
Что такое ИИ? Алан Тьюринг и его работы. Вычислительная сложность алгоритма. Идея двоичного поиска. Создание приложения, отгадывающего возраст.

Продвинутые библиотеки языка Python. Pygame (10 часов)
Обзор дополнительных библиотек для работы с графическим интерфейсом. Библиотека Pygame. Шаблон программы. Геометрические примитивы в Pygame. Простая анимация в Pygame.

События клавиатуры. События мыши. Дополнительные поверхности. Работа с текстом. Музыка.

6. Поурочное планирование

Модуль 1. Основы языка Python.

Урок 1. Вводное занятие.
Техника безопасности. Знакомство с программой курса. Краткая история языка Python, кто создал и почему так назвали. Что можно сделать, зная язык программирования Python? Где в принципе программисты могут писать программы и как они их потом запускают? Отличие компилируемых и интерпретируемых языков программирования.
Демонстрация установки интерпретатора Python. Используем Python как калькулятор: интерактивный режим работы с интерпретатором Python, вычисляем сумму чисел, вычисляем 2 в степени 100. Запуск встроенной среды разработки IDLE, работа в командной строке Shell – аналог интерактивного режима.
Тест по технике безопасности и простым фактам о языке Python.

Урок 2. Основы языка Python. Ввод-вывод данных, числа и строки, операции с числами и строками.
Встроенная среда разработки IDLE. Знакомство с функциями print() и input(). Особенность ввода данных на языке Python – данные считываются в виде строки. Первая программа: считываем число и выводим его же. Простые операции со строками, конкатенация строк. Создание простой программы, спрашивающей имя пользователя и затем приветствующей его.
Создание простой программы, спрашивающей у пользователя имя, количество лет, а затем выводящей имя столько раз, сколько пользователю лет.

Урок 3. Основы языка Python. Операции с числами и строками.
Типы данных int и str. Преобразование типов с помощью операторов int() и str(). Простые вычисления в Python, например, “Чему равно str(2 + 3) * int(‘2’ + ‘3’)? Постарайтесь дать ответ, не используя интерпретатора Python”.
Особенности функций input() и print(). Изменение поведения функции print() с помощью параметров sep и end. Символ перевода строки ‘\n’. Тест на понимание работы функций print, input, str, int.
Простые программы на взаимодействие с системой. Два возможных примера описаны ниже.

Пример 1. Поле Чудес
Программа приветствует Буратино и спрашивает, сколько у него монет, после ответа пользователя программа сообщает Буратино, сколько денег у него будет завтра.

```
print(‘Привет, Буратино!’)
print(‘Зарой все свои деньги здесь.’)
n = input(‘Сколько у тебя монет?’)
n = int(n)
print(‘Завтра у тебя будет’, n * 100, ‘монет’)
print(‘Приходи завтра!’)
```

Пример 2. Инопланетянин
Программа приветствует пользователя, узнает его возраст и печатает соответствующее количество сердечек.

```
print(‘Привет, землянин!’)
n = input(‘Сколько тебе лет?’)
n = int(n)
print(‘Шлю тебе’, n, ‘сердечек’)
print(‘\u2764’ * n)
```

Урок 4. Основы языка Python.Условная инструкция в Python.
Условная инструкция if-else в Python. Блок-схема ветвления. Неполное и полное ветвление. Отступы в Python – почему это важно? Операторы сравнения ==, !=, <, <=, >, >=, <, >.

Простые программы на использование условной инструкции. Несколько возможных примеров описаны ниже. Предложите школьникам придумать самим подобные программы и реализовать их.

Программа 1.

```
print(‘На улице темно? да/нет’)
ans = input()
if ans == ‘да’:
   print(‘спокойной ночи!’)
```

Программа 2.

```
ans = input(‘У тебя есть щупальца? да/нет’)
if ans == ‘да’:
   print(‘Привет, осьминог!’)
else:
   print(‘Привет, человек!’)
```

Урок 5. Основы языка Python.Условная инструкция в Python, elif, логические операции.
Как быть, если одновременно нужно проверить истинность нескольких условий? Или то, что верно хотя бы одно условие из нескольких? На помощь приходят логические операции or и and. Простые программы, например,

Программа 1. Пройти на аттракцион

```
rost = input(‘Каков твой рост в сантиметрах?\n’)
vozrast = input(‘Сколько тебе лет?\n’)
if int(rost) > 120 and int(vozrast) >= 3:
   print(‘Проходи!’)
else:
   print(‘Подрасти еще немного!’)
```

Если нужно больше двух ветвлений, стоит использовать оператор elif. Показать несколько простых примеров с использованием elif и без использования elif. Создание простых программ, где удобно использовать elif, например,

Программа 2. Погода

```
weather = input('Какая сегодня погода? дождь/снег/солнце')
if weather == 'дождь':
   print('Захватите зонтик!')
elif weather == 'снег':
   print('Не забудьте варежки!')
else:
   print('Ура! Берем солнечные очки!')
```

Что такое модули и как их подключать? Модуль random и некоторые его функции. Примеры использования модуля random.

Программа 3. Чье число больше?

```
import random
n = input('Загадай число от 1 до 5, и я тоже загадаю\n')
n = int(n)
m = random.randrange(1, 10)
if m > n:
   print('Мое число', m, 'больше твоего, ура!')
else:
   print('Я загадал', m, 'и проиграл...')
```

Вопрос к школьникам: модернизируйте программу так, чтобы рассматривался и случай равенства чисел.

Урок 6. Цикл for в Python.

Использование цикла для перебора объектов (конструкция for elem in object, где object – строка, кортеж, иной итерируемый объект). Использование цикла для выполнения заданного количества операций: три способа записи for i in range(n); for i in range(a, b); for i in range(a, b, d)). Простые программы, например:

- Как напечатать слово привет столько раз, какое число введет пользователь (каждый "привет" должен быть в отдельной строке)?
- Напечатайте числа от 1 до n (n определяется пользователем) в прямом и в обратном порядке, только четные, только кратные трем, и т.п.

- Пользователь задает число n, программа выводит n строк, в первой строке одна звездочка, во второй две и т.д.

Урок 7. Цикл while

Цикл с предусловием. Блок-схема. Бесконечный цикл. Оператор break для выхода из цикла. Переменная-счетчик для подсчета количества операций в цикле. Простые программы, например, такие, как приведены ниже:

Программа 1. Сладкоежка

```
candies = 0
ans = 'да'
while ans == 'да':
   candies = candies + 1
   print('Съедено конфет:', candies)
   ans = input('Хочешь еще конфетку? да/нет\n')
```

Программа 2. Бесконечный цикл:

```
while True:
   print('я буду работать вечно!')
```

Программа 3. Надоедливая программа:

```
while True:
   ans = input('я тебе не надоела? да/нет\n')
   if ans == 'да':
      print('Как некрасиво! Ухожу...')
      break
```

Программа 4. Угадай, как меня зовут?

```
print('Угадай, как меня зовут!')
ans = input()
while ans != 'python':
   print('Не угадал! Попробуй еще раз')
   ans = input()
print('Правильно, я python!')
```

Урок 8. Строки

Нумерация символов строк. Отрицательная нумерация. Функция длины строки. Перебор символов строки с помощью цикла for (два способа: for smb in str и по индексу). Срезы строк. Простые программы: напечатать заданную пользователем строку задом наперед; напечатать все символы заданной пользователем строки по одному в строке; изменить заданную пользователем строку так, чтобы символы были разделены знаком звездочка, и т.п.

Урок 9. Списки

Что такое список в Python? Нумерация элементов списка. Длина списка. Аналогия со строками. Модуль random для работы со списками, перемешивание элементов списка, выбор произвольного элемента. Простые программы, например, в списке можно хранить имена членов вашей семьи и с помощью модуля random выбирать того, кто моет посуду сегодня после ужина.

Урок 10. Функции. Встроенные функции в Python

Функции как инструмент многократного использования одного и того же кода. Локальные и глобальные переменные. Самостоятельное создание простых функций, например, функции, переводящей количество дней в количество секунд в этих днях. Некоторые встроенные функции в Python: max, min, sum, reverse, встроенные функции изменения регистра букв lower и upper.

Урок 11. Кейс 1. Создание программы-теста из нескольких вопросов.

Создание тематического теста. Программа ведет диалог с пользователем, задавая ему 1-3 вопроса, каждый по одной и той же схеме (задается вопрос, далее дается три попытки на ответ, если одна из попыток заканчивается верным ответом, выводится строка “Ответ верный!”, если ни в одной попытке не было дано верного ответа, просто показывается верный ответ.) Программа должна засчитывать верный ответ без учета регистра, для реализации этого удобно использовать встроенную

функцию lower или встроенную функцию upper.

Пример подобной программы, задающей пользователю вопрос про футбол, приведен ниже.

```
print('Тест про футбол')
ans = input('Вопрос 1. В какой стране проходил последний чемпионат мира по футболу\n')
cheker = True
attempt = 0
while cheker and attempt < 3:
    if ans.lower() == «россия» or ans.lower() == «в россии»:
        print('Ответ верный!')
        cheker = False
    else:
        if attempt < 2:
            ans = input('Попробуй еще раз!\n')
        attempt = attempt + 1
if attempt == 3:
    print('Попытки закончились... Верный ответ: Россия или В России')
```

Урок 12. Завершение работы над программой-тестом.

Добавление в тест еще одного-двух вопросов. Добавление счетчика очков (за каждый верный ответ пользователю начисляется определенное количество очков, после прохождения теста это количество очков выводится на экран). Обсуждение со школьниками того, как использование функций могло бы помочь сделать код короче (все вопросы теста однотипны), реализация при достаточном уровне подготовленности школьников.

Полезные ресурсы:

1. К. Вордерман и др. Программирование на Python: Иллюстрированное руководство для детей. Издательство: Манн, Иванов и Фербер, 2018 г.
2. Программирование для детей на языке Python. Издательство: АСТ, 2017 г.

3. Д. Бриггс. Python для детей: Самоучитель по программированию. Издательство: Манн, Иванов и Фербер, 2018 г.
4. Б. Пэйн. Python для детей и родителей. Издательство: Эксмо, 2017 г.
5. П. Томашевский. Привет, Python! Моя первая книга по программированию. Издательство: Наука и Техника, 2018 г.
6. https://pythontutor.ru/
7. https://ru.wikiversity.org/wiki/%D0%9A%D1%83%D1%80%D1%81_%D0%BF%D0%BE_%D0%B1%D0%B8%D0%B1%D0%BB%D0%B8%D0%BE%D1%82%D0%B5%D0%BA%D0%B5_Tkinter_%D1%8F%D0%B7%D1%8B%D0%BA%D0%B0_Python

Что получит ученик по окончании модуля:

- Обучающийся познакомится с основами программирования на языке Python.
- Изучит основные конструкции языка Python (условная инструкция, циклы, функции, списки, строки) на практических Напишет несложное консольное приложение.

Модуль 2. Создание приложений с помощью tkinter.

Урок 13. Знакомство с модулем tkinter.
Подключение модуля tkinter. Краткий обзор возможностей, демонстрация уже готовых приложений, разработанных с помощью tkinter. Создание простого GUI-приложения, состоящего из одного виджета, например, кнопки. Обсуждение общей последовательности действий, необходимой для создания GUI-приложения. Совершенствуем наше приложение, добавив реакцию на событие, например, щелчка левой кнопки мыши (например, виджет меняет цвет).

Урок 14. Что такое виджеты, конфигурация виджетов.
Некоторые виджеты: кнопка, текстовое поле, поле ввода, метка. Конфигурация виджетов, методы config и configure. Получение информации о состоянии виджета.

Урок 15. События в tkinter.
Как научить виджет реагировать на события – различные способы. Методы bind и unbind.

Кейс 2. Создайте интеллект-карту на тему “События в tkinter”.

Урок 16. Создание простых приложений.
Продолжаем знакомиться с tkinter на простых примерах. Создание приложения, состоящего из кнопки, подсчитывающей количество нажатий, и исчезающей после определенного количества нажатий. Создание простого приложения, состоящего из двух кнопок “Привет!” и “Пока…” и текстового поля: при нажатии на одну из кнопок в текстовом поле печатается приветствие, при нажатии на вторую кнопку, приложение закрывается.

Урок 17. Создание простых приложений.
Приложение “Радуга”: простое приложение, состоящее из кнопок, окрашенных в основные цвета радуги и текстового поля. При нажатии на кнопку, в текстовом поле появляется название цвета нажатой кнопки.

Урок 18. Создание простых приложений.
Упаковщики виджетов: pack(), grid(), place(). Создание простого калькулятора из нескольких кнопок и текстового поля для вывода результата вычислений.

Урок 19. Продвинутое рисование в tkinter.
Рисование на холсте canvas в tkinter. Создание различных геометрических фигур.

Урок 20. Движение нарисованных объектов.
Движение нарисованных объектов с помощью клавиатуры.

Урок 21. Кейс 3. Создание игрового приложения “Пинг-понг”.
Создание простой версии игры “пинг-понг”. Создаем мячик

и ракетку с помощью canvas. "Учим" ракетку двигаться влево-вправо при нажатии кнопок со стрелками влево-вправо соответственно. Как сделать так, чтобы ракетка двигалась все время и при нажатии кнопки только меняла направление движения. Как сделать так, чтобы ракетка не выходила за левую и правую границы игрового поля.

Урок 22. Создание игрового приложения "Пинг-понг".
Доделываем приложение пинг-понг. Добавляем правильное движение мячика – с отскоком от стен и от ракетки. Добавляем счетчик количества ударов мяча о ракетку.

Урок 23. Завершение работы над приложением "пинг-понг".
Работа в парах: ребята делятся на пары программист-тестер, и тестируют созданные приложения.

Общая идея игры сапер. Двумерные списки, вложенные циклы. Как расположить виджеты Label в виде прямоугольника. Раскрасим минное поле в шахматном порядке.

Урок 25. Приложение "Сапер". Информация о минах.
Двумерный битовый список для хранения информации о минах в игре. Создаем и заполняем его случайным образом. Как с помощью модуля random можно контролировать количество мин в игре? (Например, выбираем случайное целое число от 1 до 100, если оно оказалось меньшим, чем 25, в соответствующую ячейку пишем 1 – это будет мина, иначе 0 – пустая ячейка. Обсуждение со школьниками, много или мало мин в этом случае стоит ожидать).

Урок 26. Приложение "Сапер". Как узнать, сколько мин среди соседей данной клетки и как эту информацию хранить?
По созданному двумерному списку мин создаем список, в каждой ячейке которого находится информация о том, сколько мин среди соседей данной клетки (соседями считаются клетки, имеющие с данной клеткой общую вершину или сторону).

Урок 27. Создаем "мозг" игры.
Пишем "мозг" игры сапер. Что происходит в тот момент, когда пользователь нажимает на клетку, описываем реакцию виджета на событие.

Урок 28. Завершение работы над приложением "Сапер".
Доделываем приложение, тестируем, убираем недостатки.

Кейс 5. Как представить свой проект на конференции? Ребята разбиваются на группы, выбирают лучшее из созданных ими приложений, рассказывают о процессе создания.

Полезные ресурсы:

1. К. Вордерман и др. Программирование на Python: Иллюстрированное руководство для детей. Издательство: Манн, Иванов и Фербер, 2018 г.
2. Программирование для детей на языке Python. Издательство: АСТ, 2017 г.
3. Д. Бриггс. Python для детей: Самоучитель по программированию. Издательство: Манн, Иванов и Фербер, 2018 г.
4. Б. Пэйн. Python для детей и родителей. Издательство: Эксмо, 2017 г.
5. П. Томашевский. Привет, Python! Моя первая книга по программированию. Издательство: Наука и Техника, 2018 г.
6. https://pythontutor.ru/
7. https://ru.wikiversity.org/wiki/%D0%9A%D1%83%D1%80%D1%81_%D0%BF%D0%BE_%D0%B1%D0%B8%D0%B1%D0%BB%D0%B8%D0%BE%D1%82%D0%B5%D0%BA%D0%B5_Tkinter_%D1%8F%D0%B7%D1%8B%D0%BA%D0%B0_Python

Что получит ученик по окончании модуля:
- Обучающийся познакомится с встроенной библиотекой компонентов графического интерфейса Tkinter.
- Научится работать с виджетами библиотеки Tkinter, познакомится с событиями и научится описывать реакцию видже-

тов на события. Выполнит много несложных практических заданий, создавая простые GUI-приложения.
- С помощью Tkinter создаст приложения Сапер и Пинг-Понг, разберется с интеллектуальной начинкой этих игр и с тем, как ее реализовать на языке Python.

Модуль 3. Криптография

Урок 29. История криптографии. Старинные шифры
Что такое криптография и чем она занимается? Как шифровали сообщения в древности? Старинные шифры атбаш и сцитала (шифр Древней Спарты).

Урок 30. История криптографии. Старинные шифры
Чем отличается стеганография от криптографии. Диск и линейка Энея. Шифр Цезаря.

Урок 31. История криптографии. Старинные шифры
Шифрование с использованием таблиц. Квадрат Полибия. Решетка Кардано.

Урок 32. Игровое занятие. Разгадываем шифры.
Создание и расшифровка "таинственных посланий" с использованием вспомогательных материалов – карандаша и полоски бумаги для шифра считала, диска для шифра Цезаря и т.д.

Урок 33. Кейс 6. Создание криптографических приложений с помощью tkinter.
Диалоговые окна simpledialog и messagebox в tkinter. Создание приложения, запрашивающего текст для шифровки и возвращающего зашифрованное сообщение для шифра атбаш. Творческое задание: придумать свой несложный шифр перестановки и изменить приложение так, чтобы оно зашифровывало текст вашим способом (вариант шифра: переставить местами соседние буквы сообщения). Модернизация программы: пользователю сообщается название шифра, далее на выбор можно

выбрать, хочет ли пользователь зашифровать текст, или, наоборот, расшифровать его.

Урок 34. Создание криптографического приложения «Шифр Цезаря».
Продвинутое криптографическое приложение Шифр Цезаря. Создание приложения, которое по выбору пользователя может как зашифровать сообщение шифром Цезаря с данным ключом (значение ключа тоже задает пользователь), либо расшифровать сообщение, зашифрованное шифром Цезаря с произвольным ключом. Идея перебора всех возможных ключей.

Урок 35. Завершение работы над приложением.

Урок 36. Модификация и развитие шифра Цезаря.
Шифры, которые практически невозможно разгадать. Шифр Виженера.

Урок 37. Современные алгоритмы шифрования
Современные алгоритмы шифрования и основные принципы их работы. Кто и зачем придумал RSA? Идея открытого и закрытого ключа.

Урок 38. Современные алгоритмы шифрования
Еще раз обсуждение того, что такое открытый и закрытые ключи. Идея односторонней функции с лазейкой (трудновычислимой информацией). Обсуждение нескольких несложных способов шифрования с лазейкой (например, берется конкретная книга и буквы алфавита шифруются с помощью этой книги – шифром для буквы А может быть первое слово на первой странице, …., для буквы Я – тридцать третье слово на тридцать третьей странице и так далее. В данном случае лазейкой является конкретная книга, используемая для шифрования.)

Урок 39. Игровое соревновательное занятие "Взламываем шифр с открытым ключом".

Придумывание школьниками своих шифров для одноклассников и последующее разгадывание сообщений.

Урок 40. Что такое электронная подпись?
Протокол аутентификации сообщений. Протокол электронно-цифровой подписи. Алгоритмы проверки электронной подписи. Алгоритм генерации электронной подписи.

Урок 41. Кодирование текста.
Кодирование информации в компьютере. Сколько информации можно закодировать с помощью 8 бит? Таблица ASCII. Как кодируют русские буквы и почему вместо понятного текста иногда в электронных сообщениях мы видим "крокозябры". Стандартные кодовые таблицы для русского алфавита. Unicode. Вывод символа с помощью языка Python по номеру в кодовой таблице.

Урок 42. Работа с файлами в Python.
Чтение данных из файла. Методы readline(), readlines() и read(). Удаление концевых символов строки с помощью метода rstrip(). Вывод данных файл. Создание программы, считывающей данные из файла посимвольно.

Урок 43. Кейс 7. Разработка приложения, сохраняющего данные в файле.
Подготовка к созданию приложения, позволяющего открывать текстовые файлы в текстовом поле в tkinter или создавать новые, а также редактировать их и сохранять. Многострочное текстовое поле Text в tkinter. Конфигурация виджета Text (размеры, шрифт, цвет).

Урок 44. Разработка приложения, сохраняющего данные в файле.
Методы insert(), get() и delete() виджета Text. Создание простого приложения, содержащего многострочное текстовое поле и две кнопки, позволяющие добавлять текст и удалять его.

Урок 45. Разработка приложения, сохраняющего данные в файле.
Модуль filedialog в tkinter (диалоговые окна открытия и сохранения файлов). Создание приложения, позволяющего открывать текстовые файлы в текстовом поле в tkinter или создавать новые, редактировать их и сохранять. Использование полос прокрутки Scrollbar.

Урок 46. Продвинутые возможности Python.
Структура данных с идентификацией элемента не по числовому, а по произвольному ключу: словарь. Как создать словарь, как заполнить словарь, работа с элементами словаря. Возможный пример: создадим в Python словарь, в котором индексом является название страны, а значение – названием столицы этой страны. Затем добавим диалог с пользователем: программа спрашивает название страны, в котором живет пользователь, если такая страна уже есть в словаре, то выводится сообщение о том, как называется столица этой страны, в противном случае пользователю задается вопрос о том, как называется столица его страны и полученная информация сохраняется в словарь.

Урок 47. Создание продвинутого интерфейса к программе, разработанной на предыдущем уроке. Вопрос и ответ задаются и обрабатываются с помощью диалоговых окон, а информация о странах и столицах хранится в файле.

Урок 48. Завершение работы над программой.

Урок 49. Работа с датой и временем
Модуль date и datetime. Как с помощью языка Python получить сегодняшнюю дату и текущее время? Как узнать день недели для какой-то даты? Несложные практические задания, например, А.С.Пушкин родился 6 июня 1799 года. С помощью модуля datetime определите, какой был день недели. Удобное использование словаря для получения названия дня недели по номеру дня: days = {0: 'понедельник', 1: 'вторник', 2: 'среда',3:

‘четверг’,4: ‘пятница’,5: ‘суббота’,6: ‘воскресенье’}

Урок 50. Кейс 8. Создание приложения Календарь дней рождения моей семьи

Обсуждение проекта. Общая идея приложения: в файле хранится информация, например, в таком виде:
День рождения мамы,05.05.1984
День рождения папы,09.08.1983
Приложение обрабатывает эту информацию, с помощью модуля datetime и его возможностей вычисляет, сколько дней между текущей датой и днем рождения каждого члена семьи, упорядочивает данные по близости к текущей дате и выводит информацию, например, в таком виде:

Урок 51. Завершение работы над проектом.

Урок 52. Кейс 9. Разработка и создание игры “Мемори”.

Подготовка игрового поля. Использование символов Unicode в качестве картинок, например, u’\u2702’ это символ с изображением ножниц. Создание сетки из кнопок с закрытым текстом с помощью упаковщика grid. Конфигурация цвета поля и размера текста. Создание функции, открывающей текст-изображение при нажатии на кнопку. Обсуждение дальнейшей реализации.

Урок 53. Разработка и создание игры “Мемори”.

Использование словарей для хранения информации о том, какая кнопка нажата. Использование grid_info()[‘row’] и grid_info()[‘column’] для получения информации о том, в какой строке и в каком столбце находится нажатая кнопка. Модификация функции, обрабатывающей событие нажатие кнопки – как сделать так, чтобы запоминалась информация о предыдущей нажатой кнопке и обработка этой информации. Использование задержки sleep из модуля time для отображения изображений, если при двух последовательных нажатиях были открыты разные картинки.

Урок 54. Разработка и создание игры “Мемори”.

Завершение работы над проектом. Тестирование проекта и доработка мелочей (если изображения уже открыты, то на кнопки нельзя нажимать, если открыты все изображения, то игра завершена и т.п.). Тестирование проектов одноклассников. Добавление счетчика ходов. Работа с изображениями в Python. Замена текстовых надписей в приложении “Мемори” изображениями.

Работа в паре: тестируем проект напарника.

Полезные ресурсы:

1. К. Вордерман и др. Программирование на Python: Иллюстрированное руководство для детей. Издательство: Манн, Иванов и Фербер, 2018 г.
2. Программирование для детей на языке Python. Издательство: АСТ, 2017 г.
3. Введение в криптографию. Под редакцией В.В.Ященко Издание четвертое, дополненное, Москва, МЦНМО, 2012.
4. https://ru.wikipedia.org/wiki/%D0%98%D1%81%D1%82%D0%BE%D1%80%D0%B8%D1%8F_%D0%BA%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D0%B8
5. https://habr.com/ru/company/yandex/blog/324866/
6. https://tproger.ru/translations/understanding-cryptography/

Что получит ученик по окончании модуля:

- Обучающийся познакомится с историей криптографии и самыми знаменитыми шифрами.
- Потренируется решать задачи, в которых нужно разгадать зашифрованное сообщение.
- Научится создавать криптографические приложения с использованием библиотеки Tkinter.
- Получит представление о современных методах шифрования.
- Получит представление о кодировках текста и почему для русского текста существует так много кодировок.
- Научится работать с файлами в Python и создаст приложение, позволяющее сохранять информацию в файле.
- Узнает, что такое словарь, и как с ним работать в языке Python.
- Познакомится с библиотеками для работы с датой и временем и создаст с их помощью несложное приложение.
- Напишет игру «Мемори».

Модуль 4. Искусственный интеллект

Урок 55. Искусственный интеллект. История.
Что такое искусственный интеллект и что такое интеллект вообще? Есть ли IQ у компьютерных программ? Как можно сравнить человеческий и компьютерный интеллекты? Алан Тьюринг и его работы. Немного про вычислительную сложность.

Урок 56. Искусственный интеллект. Идея двоичного поиска.
Обсуждение задачи угадывания числа: учитель загадал натуральное число от 1 до 100. Ученик хочет его отгадать, и задает вопросы, на которые учитель отвечает только “да” или “нет”. За какое наименьшее количество вопросов ученик сможет отгадать загаданное число? Если перебирать числа последовательно (такой способ в программировании называется линейный поиск), за 99 вопросов точно можно угадать, но наверняка можно и за меньшее количество. Обсуждение со школьниками, школьники задают вопросы, учитель фиксирует на доске вопросы и отвечает “наихудшим” возможным образом. Обсуждение того, какие вопросы были хорошими, и почему. Обсуждение идеи сокращения количества подозрительных чисел вдвое одним вопросом. Общая формулировка идеи двоичного поиска. Обсуждение того, сколько вопросов потребуется, если загадано число от 1 до N, где N – степень двойки. Обсуждение того, почему для N = 10 трех вопросов не хватит.

Урок 57. Продолжение обсуждения идеи двоичного поиска.
Напомнить про то, что такое двоичный поиск (для того, чтобы отгадать число от 1 до 100 потребовалось бы всего 7 вопросов с ответом да/нет). Дополнительно можно обсудить, что на прошлом уроке все вопросы задавались последовательно, и каждый последующий вопрос задавался после того, как был известен ответ на предыдущий. А вот хватит ли семи вопросов, если все вопросы формулируются сразу, например, на листке, и даются учителю? (Учитель сразу дает ответы на все записанные вопросы, после чего ученик должен сказать, каким было

загаданное число). Ответ: да, семи вопросов для загаданного числа от 1 до 100 тоже хватит, например, можно спросить “какова первая цифра в двоичной записи загаданного числа, если его дополнить слева нулями до двоичного числа длины 7”, “ какова первая цифра в двоичной записи загаданного числа, если его дополнить слева нулями до двоичного числа длины 7”, и т.п.

Создание программы, отгадывающей загаданное число. Пример подобной программы перед вами.

```
from random import*
print('Я загадал число от 1 до 20. Попробуй угадать!')
secret = randint(1, 20)
ans = int(input())
while ans != secret:
    if ans > secret:
        print('Слишком большое число!')
    else:
        print('Маловато...')
    ans = int(input('Попробуй еще раз!\n'))
print('Молодец, угадал!')
```

Предложите школьникам модернизировать программу так, чтобы загадывалось число от 1 до n, где n задает пользователь. Кроме того, пусть программа подсчитывает количество заданных вопросов и после того, как число отгадано, сообщает это количество.

Урок 58. Кейс 10: приложение, угадывающее возраст.

Создание приложения на языке Python, отгадывающее возраст пользователя. Идея: допустим, человек родился 16 июля. Умножим день рождения на 2, получим 32 (16*2). Прибавим к результату 5, получим 37 (16*2+5). Умножим результат на 50, получим 1850 ((16*2+5)*50=16*100+250). К результату прибавим номер месяца рождения, получим 1857. Результат вычислений попросим сообщить нам. Для того, чтобы узнать дату рождения, достаточно вычесть из результата вычислений число 250,

получится число 1607, первые две его цифры соответствуют дню, последние две – месяцу.

Для удобства реализации создадим словарь, ключами в котором являются числовые номера месяцев, а значениями – названия месяцев. Пример возможной реализации приведен ниже.

```
D = {1: 'января', 2:'февраля', 3:'марта', 4:'апреля', 5:'мая', 6:'июня',
7:'июля', 8:'августа', 9:'сентября', 10:'октября', 11:'ноября', 12:'декабря'}
input('Привет! Я отгадаю дату твоего рождения. Нажми enter, как будешь готов.')
input('Умножь день, в который ты родился, на 2. Если ты родился 15 февраля, умножай на 2 число 15. Нажми enter, как будешь готов. ')
input('К результату прибавь 5. Нажми enter, как будешь готов.')
input('Полученное число умножь на 50. Нажми enter, как будешь готов.')
input('К результату прибавь номер месяца, в который ты родился (число от 1 до 12). Нажми enter, как будешь готов.')
ans = int(input('А теперь скажи, что у тебя получилось.'))
ans = ans - 250
print('Ты родился', ans//100, D[ans % 100])
```

Что получит ученик по окончании модуля:

- Обучающийся познакомится с понятием искусственного интеллекта и с историческими фактами, касающимися искусственного интеллекта.
- Разберется с идеей двоичного поиска.
- Напишет несколько простых консольных приложений.

Модуль 5. Продвинутые библиотеки языка Python. Pygame.

Урок 59. Знакомство с продвинутыми фреймворками для разработки GUI-приложений.

Обзор дополнительных библиотек, позволяющих создавать приложения с графическим интерфейсом (PyQt, wxPython,

Pygame). Примеры приложений. Установка библиотеки Pygame.

Урок 60. Шаблон программы на Pygame
Как устроен шаблон программы на Pygame. Подключение модулей, инициализация, главный цикл, корректное завершение работы приложения. Функции init(), set_mode(), update(), quit(). События.

Урок 61. Геометрические примитивы в Pygame
Цветовые модели. Рисуем и раскрашиваем линии, прямоугольники, многоугольники, круги, овалы.

Урок 62. Кейс 11. Простая анимация в Pygame.
Создание анимации движения геометрической фигуры от левой границы главного окна вправо. Как изменить программу так, чтобы фигура не исчезала, а отражалась от стен.

Урок 63. События клавиатуры.
Модуль pygame.event.

Кейс 12: создание приложения, реализующего перемещение геометрической фигуры с помощью стрелок вправо/влево и вверх/вниз.

Урок 64. События мыши.
Обработка событий нажатия кнопки мыши, отпускания кнопки мыши, движения мыши. Координаты мыши.

Кейс 13: "Звездное небо". Создание приложения, в котором при щелчке мыши по экрану на экране появляются звездочка.

Урок 65. Дополнительные поверхности в Pygame.
Surface и blit(). Простые примеры.

Урок 66. Кейс 14. Работа в команде. Создание приложения "Поймай звезду".

Создание игры, в которой в произвольном месте экрана появляются звездочки, падающие вниз. При щелчке мыши по звездочке она исчезает, а в соответствующем текстовом поле идет подсчет пойманных звезд. Классы и объекты. Реализация появления звезд. Ребята разбиваются на группы, распределяют задачи внутри группы.

Урок 67. Продолжение работы над игрой. Текст в Pygame.
Реализация реакции звезды на событие щелчка мыши. Работа с текстом в Pygame. Модуль pygame.font.

Урок 68. Завершение работы над проектом. Музыка в Pygame.
Добавление фоновой музыки и звуковых эффектов в игру. Модули pygame.mixer и pygame.mixer.music.
Представление проекта.

Полезные ресурсы:
1. https://younglinux.info/pygame/pygame
2. https://habr.com/ru/post/347138/

Что получит ученик по окончании модуля:
- Обучающийся получит представление о некоторых дополнительных библиотеках языка Python, позволяющих разрабатывать приложения с GUI (PyQt, wxPython, Pygame).
- Обучающийся познакомится с принципами работы основных элементов библиотеки Pygame и научится понимать код, использующий Pygame.
- В процессе обучения ученик разработает и напишет игровое приложение с использованием Pygame.

7. Планируемые результаты обучения.

Важнейшими умениями/знаниями являются следующие:

- умение пользоваться персональным компьютером и его периферийным оборудованием;
- умение следовать требованиям техники безопасности, гигиены, эргономики и ресурсосбережения при работе со средствами информационных и коммуникационных технологий;
- умение осуществлять взаимодействие посредством электронной почты, чата, форума;
- умение искать информацию с применением правил поиска (построения запросов), в компьютерных сетях, некомпьютерных источниках информации (справочниках и словарях, каталогах, библиотеках) при выполнении заданий и проектов по различным учебным дисциплинам;
- знакомство с основными конструкциями языка Python (условная инструкция, циклы, функции, списки, строки) на практических примерах;
- умение работать со встроенной библиотекой компонентов графического интерфейса tkinter.
- формирование представления о некоторых дополнительных библиотеках языка Python, позволяющих разрабатывать приложения с GUI (PyQt, wxPython, Pygame);
- формирование представления о современных методах шифрования;
- знакомство с понятием искусственного интеллекта и с историческими фактами, касающимися искусственного интеллекта;
- формирование умений разрабатывать несложные консольные приложения и приложения с графическим интерфейсом;
- умение выбирать способ представления своего проекта с использованием соответствующих программных средств.

www.roskvantorium.ru/fond

Фонд новых форм
развития образования
PLUS ULTRA | ДАЛЬШЕ ПРЕДЕЛА