



МИНИСТЕРСТВО ПРОСВЕЩЕНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ



Фонд новых  
форм развития  
образования

**ТОЧКА РОСТА**  
ФЕДЕРАЛЬНАЯ СЕТЬ ЦЕНТРОВ  
РАЗВИТИЯ ОБРАЗОВАНИЯ  
И УЧАЩЕГОСЯ ПРОФИЛЕЙ

**Муниципальное казённое общеобразовательное учреждение  
«Новокрестьяновская средняя общеобразовательная школа»**

**Центр образования цифрового и гуманитарного профилей  
«Точка роста»**

«Согласовано»

Руководитель центра  
«Точка роста»

*Мо* Моисеева А. Л.

от «29» 08 20 22 г.



«Утверждено»

Директор МКОУ

«Новокрестьяновская СОШ»

*Мант* Мансурова Т.М.

Приказ № 127  
от «29» 08 20 22 г.

**РАБОЧАЯ ПРОГРАММА**

**занятий дополнительного образования**

**«Основы Python + tkinter»**

**Составитель:  
педагог по информатике  
Бабенко Д.М.**

с. Новокрестьяновское  
2022- 2023 учебный год

## 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 два блока:

- Основы языка Python

- Создание приложений с помощью tkinter

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: фронтальные, 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 36 часов. Рабочая программа рассчитана на 36 учебные недели, 1 час в неделю, общее количество часов — 36. Рабочая программа может реализовываться использованием электронного обучения (ЭО) и дистанционно 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 процессе 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

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

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 “пингпонг”, “сапер”.

## 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 выбрать 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')

checker = True

attempt = 0

while checker and attempt < 3:

    if ans.lower() == «россия» or ans.lower() == «в россия»:

        print('Ответ верный!')

        checker = 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](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.

Уроки 29-36. Практические работы

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](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 виджетов 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,

Материально-техническая база:

- МФУ (принтер, сканер, копир) «Lexmark»
- Ноутбук учителя «ACER»
- Мобильный класс «ACER»
- Интерактивный комплекс «Nextouch»
- Мобильное крепление для интерактивного комплекса.
- Вычислительный блок интерактивного комплекса
- Планшет « Apple»
- Фотоаппарат с объективом «Canon»
- Штатив
- Карта памяти для фотоаппарата/видеокамеры