



### Методические рекомендации по теме

«Разбиение задачи на подзадачи с использованием функций. Проект «Загадки»»

#### Цель:

- дать представление о прикладном использовании функций в языке Python.

#### Задачи:

- практика применения функций в Python;
- разработка программного проекта «Загадки» в Python;
- анализ программного кода с целью определения, что выведет программа при конкретных исходных данных;
- исправление ошибок и дописывание программного кода;
- написание программного кода.

#### Планируемые результаты

*Личностные:* обучающиеся получают навыки активной коммуникации в группе, осознанной ориентировки в мире ИТ профессий, постановки собственных образовательных целей и задач, владение первичными навыками анализа и критичной оценки получаемой информации.

*Предметные:* обучающиеся получают представления об использовании функций в Python при разработке реального программного продукта.

*Метапредметные:* обучающиеся получают возможность владения общепредметными понятиями «функция», «пользовательская функция», «аргумент функции»; информационно-логическими умениями; умениями самостоятельно планировать пути достижения целей; владения умениями принятия решений и осуществления осознанного выбора;

повышения уровня ИКТ – компетентности и расширение кругозора в области информатики и программирования; знакомство с профессиональной деятельностью программиста в рамках ранней профориентации; развитие интеллектуальных способностей, а также логического и критического мышления.

## **Материалы к занятию**

Приложение 1: Сценарный план видеоролика

Приложение 2: Домашнее задание и практика

Приложение 3: Краткие организационно-методические рекомендации по организации работы на занятии

Приложение 4: Рекурсия (дополнительно)

## **Ход проведения урока**

### **1. Организационный момент.**

**Мотивация на учебную деятельность.**

Приветствие учащихся, сообщение темы и целей занятия.

### **2. Вводный блок.**

**Тема.**

Преподаватель при необходимости останавливая трансляцию, комментируя дополнительно тему занятия.

**Проблемная дискуссия** по вопросам:

- Какие важные признаки есть у загадки (загадка и ответ)?
- Как можно использовать пользовательские функции если мы планируем делать проект по отгадыванию загадок?
- Как может выглядеть и работать такой проект?
- Попросите учеников придумать и записать несколько небольших загадок для последующей работы над проектом

**Итоги дискуссии** (обобщаются преподавателем и фиксируются ответы учеников на доске, чтобы вернуться к ним и оценить правильность предположений учеников на этапе рефлексии):

- Идеи по внешнему виду и механике работы проекта
- Примеры нескольких загадок

*\*см. сцены 1 – 2 (здесь и далее приводится Таблица «Содержание видеоролика». Приложение 1).*

### **3. Блок повторения.**

#### **Блиц-опрос.**

Преподаватель предлагает ученикам ответить на **5 вопросов** по предыдущей теме; задания выполняются в сопровождении видеоролика с использованием таймера; ученики выполняют задания, голосуют, обсуждают результаты. Процедура голосования определяется инструкцией **в сцене 3**; учитель должен убедиться, что всем понятна процедура голосования. *Преподаватель может поставить ролик на паузу и обсудить результаты голосования; объяснить правильный ответ руководствуясь материалами предыдущего занятия*

*\*см. сцены 3 – 7*

### **4. Теоретический блок.**

#### **Структура проекта с пользовательскими функциями.**

Новый материал излагается в сопровождении видеоролика, рекомендуется разместить на доске или флип-чарте изображения объектов, сопровождающих материалы по теме.

Обсуждением вопросов по просмотренным материалам:

- Какую структуру будет иметь наш проект?
- Как создается сложный список?

- Для чего мы подключим модуль random?

*При необходимости преподаватель может поставить ролик на паузу и дать дополнительные пояснения по материалу; если ответы на вопросы вызывают у учеников затруднения, преподаватель может вывести нужную сцену ролика на экран для помощи ученикам.*

*\*см. сцена 8, 15, 20-21*

## **5. Блок заданий.**

**Проект: «Загадки».**

**«Загадки»:** включает одно практическое задание, которое выполняется в несколько этапов:

- Создание базы данных с загадками и ответами,
- Создание цикла для вывода загадки и получение ответа
- Создание функции, которая будет определять правильность ответа
- Создание счетчика правильных ответов

*На сцене разбора задания преподаватель ставит ролик на паузу и вместе с учениками проводит разбор задания.*

*\*см. сцены 9 – 19 (кроме сцен с теорией).*

## **6. Рефлексия. Сообщение домашнего задания.**

Завершаем демонстрацией ролика и кратким обобщением материалов занятия. Преподаватель возвращается к зафиксированным в ходе дискуссии в начале урока предположениям учеников и обсуждает насколько их предположения были правильными, делаются выводы.

Преподаватель дает ученикам домашнее задание к следующему занятию (*Приложение 2*).

*\*см. сцена 22*

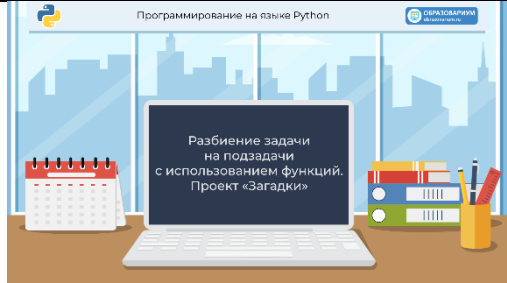
## Сценарный план видеоролика










В таблице «Содержание видеоролика» представлены:

- название блоков видеоролика (тайминг);
- краткое описание содержания в каждом блоке;
- фрагменты из видеоролика, относящиеся к соответствующему блоку;
- номера сцен в каждом блоке.

*Учитель при подготовке к уроку может ознакомиться с содержанием видеоролика в текстовом формате, при необходимости распечатать фрагменты текста или примеры заданий и задач для использования в работе с учениками. Распечатанные тексты и задания из таблицы также можно применять в качестве раздаточного материала как на уроке, так и для домашних заданий.*

Таблица. Содержание видеоролика

Название блока	Содержание блока и комментарии	Фрагменты из видеоролика	№ сцен
Вводный блок. Мы узнаем	Обозначаем ученикам тему и цели урока.  Разбиение задачи на подзадачи с использованием функций. Проект «Загадки»	 <p>Сцена 1</p>	1 2

	<p>Мы уже познакомились с таким понятием как <b>пользовательская функция</b>. Это часть кода, которая предназначена для неоднократного использования, расположена отдельно от основной программы, и объявляется при помощи команды <b>def</b>.</p> <p>Нам предстоит сделать проект, в котором сначала создадим, а потом используем наши пользовательские функции.</p>	<div data-bbox="1512 231 1960 263">  Разделение задачи на подзадачи с использованием функций. Проект «Загадки» <div data-bbox="1881 231 1960 255">  Олимпиада </div> </div> <div data-bbox="1512 319 1758 430"> <div data-bbox="1512 319 1758 343"> Пользовательская функция </div> <div data-bbox="1534 359 1736 406"> Часть кода, которая предназначена для использования в нескольких местах программы и объявляется при помощи команды def. </div> </div> <div data-bbox="1825 295 1937 470">  </div> <div data-bbox="1489 510 1568 542"> Сцена 2 </div>	
<p>Блок повторения.</p> <p><b>Блиц-опрос</b></p>	<p><i>Повторение материала предыдущего урока; на столе имеются пронумерованные карточки; после каждого вопроса выбираем ту, номер которой, совпадает с правильным ответом.</i></p> <p><b>Первый вопрос.</b> Укажите строку с правильной записью</p> <ol style="list-style-type: none"> <li>1) def имя_функции[ ]:</li> <li>2) def имя_функции{ }:</li> <li>3) def имя_функции( ):</li> <li>4) def имя_функции:</li> </ol> <p><i>Ответ 3. Правильно записывать так: <b>def имя_функции( ):</b></i></p>	<div data-bbox="1512 582 1960 614">  Разделение задачи на подзадачи с использованием функций. Проект «Загадки» <div data-bbox="1881 582 1960 606">  Олимпиада </div> </div> <div data-bbox="1512 670 1758 813"> <div data-bbox="1512 670 1758 694"> Блиц-опрос </div> <div data-bbox="1534 694 1736 710"> Поднимайте карточки с правильными ответами. </div> <div data-bbox="1534 710 1691 726"> Вопрос № 1 </div> <div data-bbox="1534 726 1691 742"> Укажите строку с правильной записью. </div> <div data-bbox="1534 742 1691 790"> 1) def имя_функции[ ]: 2) def имя_функции{ }: 3) def имя_функции( ): 4) def имя_функции: </div> </div> <div data-bbox="1702 622 1892 654"> Правильно записывать так: <b>def имя_функции( ):</b> </div> <div data-bbox="1803 662 1948 790">  </div> <div data-bbox="1489 861 1568 893"> Сцена 3 </div>	<p>3</p> <p>4</p> <p>5</p> <p>6</p> <p>7</p>
	<p><b>Второй вопрос.</b> В каком варианте правильно вызвана следующая функция?</p> <p><b>def func(arg):</b></p> <ol style="list-style-type: none"> <li>1) func()</li> <li>2) func(10)</li> <li>3) func[arg]</li> <li>4) func.arg()</li> </ol> <p><i>Ответ 2. Функция создана с одним аргументом, значит в скобках должны быть указаны данные. Правильная запись вызова может быть такой: <b>func(10)</b>.</i></p>	<div data-bbox="1512 965 1960 997">  Разделение задачи на подзадачи с использованием функций. Проект «Загадки» <div data-bbox="1881 965 1960 989">  Олимпиада </div> </div> <div data-bbox="1512 1053 1758 1197"> <div data-bbox="1512 1053 1758 1077"> Блиц-опрос </div> <div data-bbox="1534 1077 1736 1093"> Поднимайте карточки с правильными ответами. </div> <div data-bbox="1534 1093 1691 1109"> Вопрос № 2 </div> <div data-bbox="1534 1109 1736 1125"> В каком варианте правильно вызвана следующая функция? </div> <div data-bbox="1601 1125 1668 1141"> def func(arg): </div> <div data-bbox="1534 1141 1691 1189"> 1) func() 2) func(10) 3) func[arg] 4) func.arg() </div> </div> <div data-bbox="1646 1005 1937 1045"> Функция создана с одним аргументом, значит в скобках должны быть указаны данные. Правильная запись вызова может быть такой: <b>func(10)</b> </div> <div data-bbox="1803 1045 1948 1173">  </div> <div data-bbox="1489 1244 1568 1276"> Сцена 4 </div>	

**Третий вопрос.** Как называются переменные, которые созданы внутри функции?

- 1) функциональные
- 2) локальные
- 3) местные
- 4) глобальные

**Ответ 2.** Переменные, которые созданы внутри функции, называются **локальными**.

**Четвертый вопрос.** В какой строке создан словарь?

- 1) {key, value}
- 2) {key: value}
- 3) (key, value)
- 4) [key, value]

**Ответ 2.** Правильная запись словаря такая: **{key: value}**

**Пятый вопрос.** Какие индексы у числа «10»?

[[1,2,3], [4, 5, 6], [7,8,9,10]]

Поднимите карточки с соответствующими числами.

**Ответ 2 и 3.** Индексы числа 10 – **[2][3]**

[[1, 2, 3], [4, 5, 6], [7, 8, 9, 10]]

0
1
2

Разделение задачи на подзадачи с использованием функций. Проект «Загадки»

Правильный ответ: Переменные, которые созданы внутри функции, называются **локальными**.

**Блиц-опрос**  
Поднимайте карточки с правильными ответами.

Вопрос № 3  
Как называются переменные, которые созданы внутри функции?

1) функциональные  
2) локальные  
3) местные  
4) глобальные

**Сцена 5**

Разделение задачи на подзадачи с использованием функций. Проект «Загадки»

Правильная запись для словаря такая: **{key: value}**

**Блиц-опрос**  
Поднимайте карточки с правильными ответами.

Вопрос № 4  
В какой строке создан словарь?

1) {key, value}  
2) {key: value}  
3) (key, value)  
4) [key, value]

**Сцена 6**

Разделение задачи на подзадачи с использованием функций. Проект «Загадки»

Правильный ответ: Индексы числа 10 – **[2][3]**


**Блиц-опрос**  
Поднимайте карточки с правильными ответами.

Вопрос № 5  
Какие индексы у числа «10»?

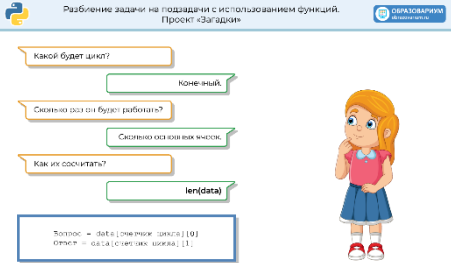
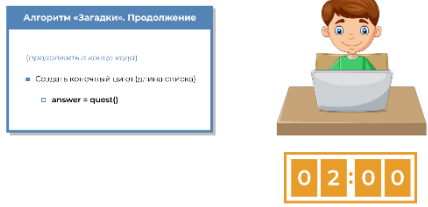
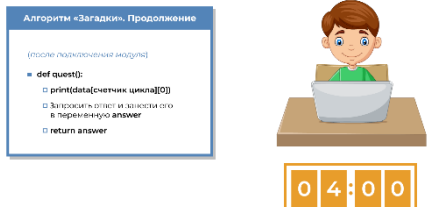
[[1, 2, 3], [4, 5, 6], [7, 8, 9, 10]]

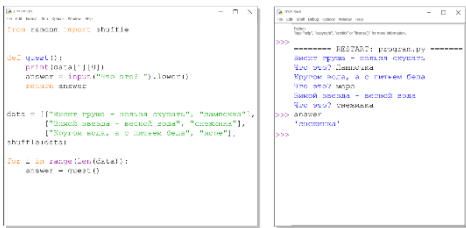
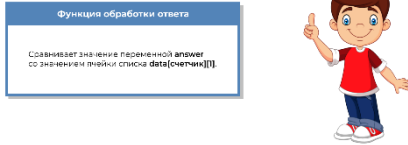
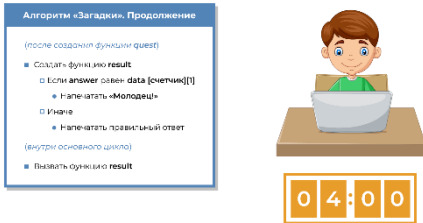
Поднимайте карточки с соответствующими числами.

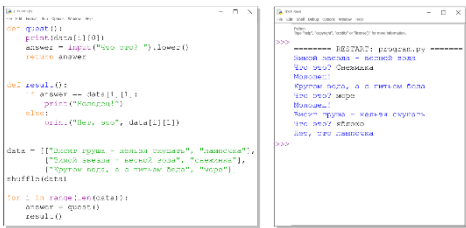
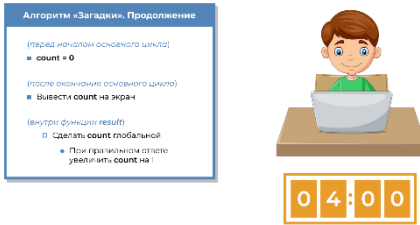
**Сцена 7**

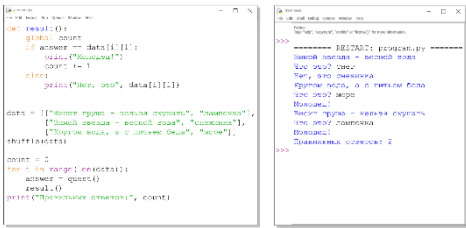
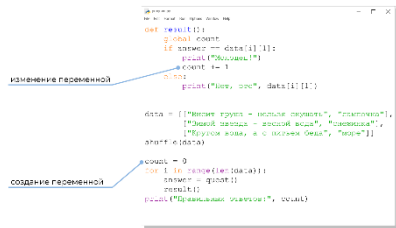
<p>Теоретический блок.</p> <p><b>Структура проекта</b></p>	<p>Наш проект будет посвящен загадкам, мы его по всем правилам – с использованием <b>пользовательских функций</b>.</p> <p>Создадим базу данных, в которой будет несколько загадок и ответов на них. В этом нам поможет структура, которая называется <b>сложный список</b>.</p> <p>Также мы создадим две пользовательские функции: одна будет задавать вопрос, вторая анализировать ответ. Игровой цикл будет вызывать данные функции.</p> <p>Вспомним, что из себя представляет <b>сложный список</b>. Это структура, в которой каждая ячейка тоже является списком. В нашем случае это будет по две ячейки – загадка и ответ. Чтобы перемешать ячейки списка, подключим модуль <b>random</b>.</p>	<div><div>Разделение задачи на подзадачи с использованием функций. Проект «Загадки»</div><div><div>Структура проекта</div><ul style="list-style-type: none"><li>Подключение модуля</li><li>Функция вопрос</li><li>Функция ответ</li><li>Сложный список с данными</li><li>Игровой цикл: Выход функции вопрос, вызов функции ответ</li></ul></div><div><div>Сложный список</div><p>это структура, в которой каждая ячейка тоже является списком.</p><pre>[ [загадка 1, ответ 1], [загадка 2, ответ 2], [загадка 3, ответ 3], ... ]</pre></div></div> <p>Сцена 8</p>	<p>8</p>
<p>Блок заданий.</p> <p><b>Практические задания:</b></p> <p>Задание 1</p>	<p><i>После окончания дикторского текста запускается таймер на 5 мин.</i></p> <p><b>Задание 1. Проект «Загадки».</b></p> <ul style="list-style-type: none"><li>Подключить <b>random</b></li><li><code>data = [</code> <code>["Висит груша - нельзя скушать", "лампочка"],</code> <code>...</code> <code>]</code></li><li><b>shuffle(data)</b></li></ul>	<div><div>Разделение задачи на подзадачи с использованием функций. Проект «Загадки»</div><div><div>Алгоритм «Загадки»</div><ul style="list-style-type: none"><li>Подключить random</li><li><code>data = [</code> <code>["Висит груша - нельзя скушать", "лампочка"],</code> <code>...</code> <li><code>shuffle(data)</code></li></li></ul></div><div></div></div> <p>Сцена 9</p>	<p>9 10 11 12 13 14</p>
	<p><b>Разбор задания 1.</b> Давайте сравним код и запустим его.</p> <p>Если ошибок не найдено – значит всё составлено верно.</p> <p>В первой строке вместо звездочки можно сразу указать команду <b>shuffle</b>.</p> <p>Таким образом программа занимает меньше оперативной памяти</p>	<div><div>Разделение задачи на подзадачи с использованием функций. Проект «Загадки»</div><div><div>Код программы</div><pre>import random from random import shuffle  data = [ ["висит груша - нельзя скушать", "лампочка"],         ["висит груша - нельзя скушать", "лампочка"],         ["висит груша - нельзя скушать", "лампочка"],         ["висит груша - нельзя скушать", "лампочка"] ]  shuffle(data)</pre></div><div><div>Выход программы</div><pre>&gt;&gt;&gt; &gt;&gt;&gt; &gt;&gt;&gt;</pre></div></div> <p>Сцена 10</p>	

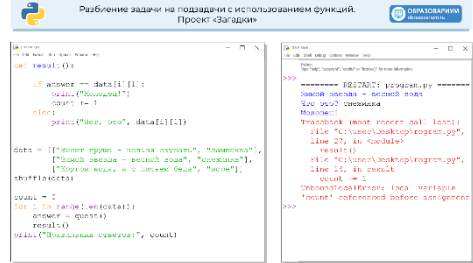
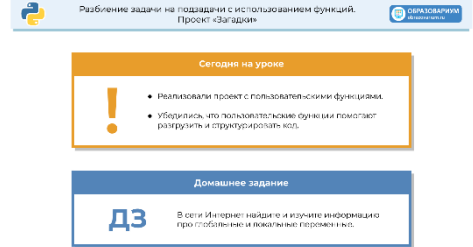


	<p>Теперь приступим к созданию игрового цикла и функций.</p> <p>Поскольку количество вопросов нам известно, следовательно потребуется конечный цикл, который проработает столько раз – сколько у нас загадок. Точнее – сколько основных ячеек в нашем сложном списке. А посчитает нам это команда <b>len</b>. Также создадим первую функцию, которая будет задавать вопрос. Поскольку наш список <b>сложный</b>, то ячейка с вопросом или с ответом имеет два индекса – <b>внешний</b> и <b>внутренний</b>. И если внешний нам будет давать переменная счетчик из цикла, то внутренний индекс вопроса всегда 0, а у ответа всегда 1.</p>	 <p>Сцена 11</p>	
	<p><i>После окончания дикторского текста запускается таймер на 2 мин.</i></p> <p><b>Задание 1. Проект «Загадки». Продолжение</b></p> <p><i>(продолжить в конце кода)</i></p> <ul style="list-style-type: none"> <li>Создать конечный цикл (длина списка) <ul style="list-style-type: none"> <li><code>answer = quest()</code></li> </ul> </li> </ul>	 <p>Сцена 12</p>	
	<p><i>После окончания дикторского текста запускается таймер на 4 мин.</i></p> <p><b>Задание 1. Проект «Загадки». Продолжение</b></p> <p><i>(после подключения модуля)</i></p> <ul style="list-style-type: none"> <li><b>def quest():</b> <ul style="list-style-type: none"> <li><code>print(data[счетчик_цикла][0])</code></li> <li>запросите ответ и занесите его в переменную <code>answer</code></li> <li><b>return</b> <code>answer</code></li> </ul> </li> </ul>	 <p>Сцена 13</p>	

	<p><b>Разбор задания 1.</b> Ваш код может выглядеть так.</p> <pre> from random import shuffle  def quest():     print(data[i][0])     answer = input("Что это? ").lower()     return answer data = ["Висит груша - нельзя скушать", "лампочка"],       ["Зимой звезда - весной вода", "снежинка"],       ["Разноцветное коромысло через реку повисло", "радуга"]] shuffle(data) for i in range(len(data)):     answer = quest() </pre>	 <p>Сцена 14</p>	
	<p>Наш следующий шаг - создание функции, которая будет определять правильность ответа. Она должна сравнить значение переменной <b>answer</b> с ячейкой списка, у которой внутренний индекс равен 1.</p> <p><b>Функция обработки ответа</b> Сравнивает значение переменной <b>answer</b> со значением ячейки списка <b>data[счетчик][1]</b>.</p>	 <p>Сцена 15</p>	15
<p>Блок заданий. <b>Практические задания:</b> Задание 1. продолжение</p>	<p><i>После окончания дикторского текста запускается таймер на 4 мин.</i></p> <p><b>Задание 1. Проект «Загадки». Продолжение</b> (после создания функции <i>quest</i>)</p> <ul style="list-style-type: none"> <li>Создать функцию <b>result</b> <ul style="list-style-type: none"> <li>Если <b>answer</b> равен <b>data [счетчик ] [1]</b></li> </ul> </li> <li>Напечатать «Молодец!» <ul style="list-style-type: none"> <li>Иначе</li> </ul> </li> <li>Напечатать правильный ответ (внутри основного цикла) <ul style="list-style-type: none"> <li>Вызвать функцию <b>result</b></li> </ul> </li> </ul>	 <p>Сцена 16</p>	16 17

	<p><b>Разбор задание 1.</b> Код программы будет выглядеть так:</p> <pre> from random import shuffle  def quest():     print(data[i][0])     answer = input("Что это? ").lower()     return answer  def result():     if answer == data[i][1]:         print("Молодец!")     else:         print("Нет, это", data[i][1]) data = [     ["Висит груша - нельзя скушать", "лампочка"],     ["Зимой звезда - весной вода", "снежинка"],     ["Разноцветное коромысло через реку повисло", "радуга"] ] shuffle(data) for i in range(len(data)):     answer = quest()     result() </pre>	 <p>Сцена 17</p>	
<p>Блок заданий. <b>Практические задания:</b> Задание 1. Продолжение</p>	<p><i>После окончания дикторского текста запускается таймер на 4 мин.</i></p> <p><b>Задание 1. Проект «Загадки». Продолжение</b> (перед началом основного цикла)</p> <ul style="list-style-type: none"> <li>count = 0</li> </ul> <p>(после окончания основного цикла)</p> <ul style="list-style-type: none"> <li>Вывести count на экран</li> </ul> <p>(внутри функции <b>result</b>)</p> <ul style="list-style-type: none"> <li>Сделать count глобальной             <ul style="list-style-type: none"> <li>При правильном ответе увеличить count на 1</li> </ul> </li> </ul>	 <p>Сцена 18</p>	<p>18 19</p>

	<p><b>Разбор задание 1.</b> Код программы будет выглядеть так:</p> <pre> from random import shuffle  def quest():     print(data[i][0])     answer = input("Что это? ").lower()     return answer  def result():     global count     if answer == data[i][1]:         print("Молодец!")         count += 1     else:         print("Нет, это", data[i][1]) data = [     ["Висит груша - нельзя скушать", "лампочка"],     ["Зимой звезда - весной вода", "снежинка"],     ["Разноцветное коромысло через реку повисло", "радуга"] ] shuffle(data) count = 0 for i in range(len(data)):     answer = quest()     result() print("Правильных ответов:", count) </pre>	 <p>Сцена 19</p>	
<p>Теоретический блок.</p> <p><b>Глобальные и локальные переменные</b></p>	<p>Может возникнуть вопрос: почему переменную count мы сделали глобальной, а остальные (answer, счетчик) – нет?</p> <p>Дело в том, что переменная count создана в основной программе, но внутри функции она меняет свое значение. А остальные переменные внутри функций остаются неизменными.</p>	 <p>Сцена 20</p>	<p>20 21</p>

	<p>Поэтому данной переменной требуется особое разрешение и если убрать строку с командой <b>global</b>, запустить код и дать правильный ответ, будет сообщение об ошибке.</p> <p>UnboundLocalError: local variable 'count' referenced before assignment</p>	 <p>Сцена 21</p>	
<p>Блок завершения занятия.</p> <p><b>Рефлексия.</b></p> <p><b>Сообщение домашнего задания</b></p>	<p><i>Завершаем демонстрацией ролика и кратким обобщением материалов занятия.</i></p> <p><b>Подведем итоги.</b></p> <p>Мы узнали:</p> <ul style="list-style-type: none"> <li>как реализовать проект с пользовательскими функциями.</li> <li>что пользовательские функции помогают разгрузить и структурировать код.</li> </ul> <p><i>Преподаватель дает ученикам домашнее задание к следующему занятию (Приложение 2).</i></p>	 <p>Сцена 22</p>	22

## Приложение 2

### Домашнее задание

Найдите в различных источниках (интернет, литература) информацию по использованию глобальных и локальных переменных в языке Python. Проанализируйте и изучите данную информацию.

*Задание можно выполнить на компьютере и представить результат и код в виде файла или снимка экрана, или распечатки.*

### Практика

Проект «Камень, ножницы, бумага»

Создайте функцию для выбора фигуры игроком (камень, ножницы или бумага). Функция должна возвращать выбранную фигуру.

Создайте функцию для выбора фигуры компьютером. Фигура должна выбираться случайным образом. Функция должна возвращать выбранную фигуру.

Создайте функцию для анализа комбинации выбора игрока и компьютера. Функция должна выводить на экран результат (победа или проигрыш).

Используя созданные функции напишите основной код игры. В качестве дополнения вы можете сделать проект многократным (добавив функции в цикл), а также продумать систему подсчета количества побед.

### Приложение 3

#### **Краткие организационно-методические рекомендации по организации работы на занятии**

«Разбиение задачи на подзадачи с использованием функций. Проект «Загадки»».

**В начале занятия** необходимо повторить материал предыдущего занятия: что такое пользовательская функция, где создается, как вызывается, что может делать, что такое аргументы, в чем разница между локальной и глобальной переменной.

**Перед просмотром блока повторения** из ролика необходимо раздать дидактический материал для выполнения заданий из блока повторения (по 4 пронумерованных карточки)

Во время голосований карточками можно останавливать ролик и вести учет правильных ответов. По окончании блока – отметить тех, у кого наилучший результат. Далее карточки необходимо собрать.

**Перед началом проекта** рекомендую набросать список коротких загадок, которые будут использоваться в проекте. Чем их будет больше – тем лучше. Также отдельно можно проговорить индексы ячеек сложного списка, которые будут

являться вопросами и ответами. Игровой проект состоит из четырех разделов, однако основной код содержится в первых трех. Так что если не будет хватать времени, то последним разделом можно пожертвовать. После создания очередного раздела не забывайте проверять код на работоспособность. Особое внимание уделите тому, что код пишется в разных местах: то в начале, то в середине проекта, то внутри цикла, то внутри функции.

Также обратите внимание на разные способы вызова функций. Это связано с тем, что функция **quest** возвращает нам результат своей работы, который мы должны куда-то принять.

## *Приложение 4: Рекурсия*

### **Рекурсия**

В программировании есть ряд задач, которые могут быть решены с помощью рекурсии.

Рекурсивная функция – это такая функция, которая вызывает сама себя. При каждом очередном вызове функция использует данные, созданные во время предыдущего вызова.

Рассмотрим работу рекурсивной функции на классическом примере нахождения факториала. Сравним программные коды примера 1 и примера 2.

Факториал числа равен произведению всех натуральных чисел от 1 до данного числа.

## Пример 1

```
program.py
File Edit Format Run Options Window Help

def fact1(n):
    factorial = 1
    for i in range(1, n + 1):
        factorial = factorial * i
    return factorial

number = int(input('Введи число: '))
print(fact1(number))
```

## Пример 2

```
program.py
File Edit Format Run Options Window Help

def fact2(n):
    if n == 1:
        return 1
    else:
        return n * fact2(n - 1)

number = int(input('Введи число: '))
print(fact2(number))
```

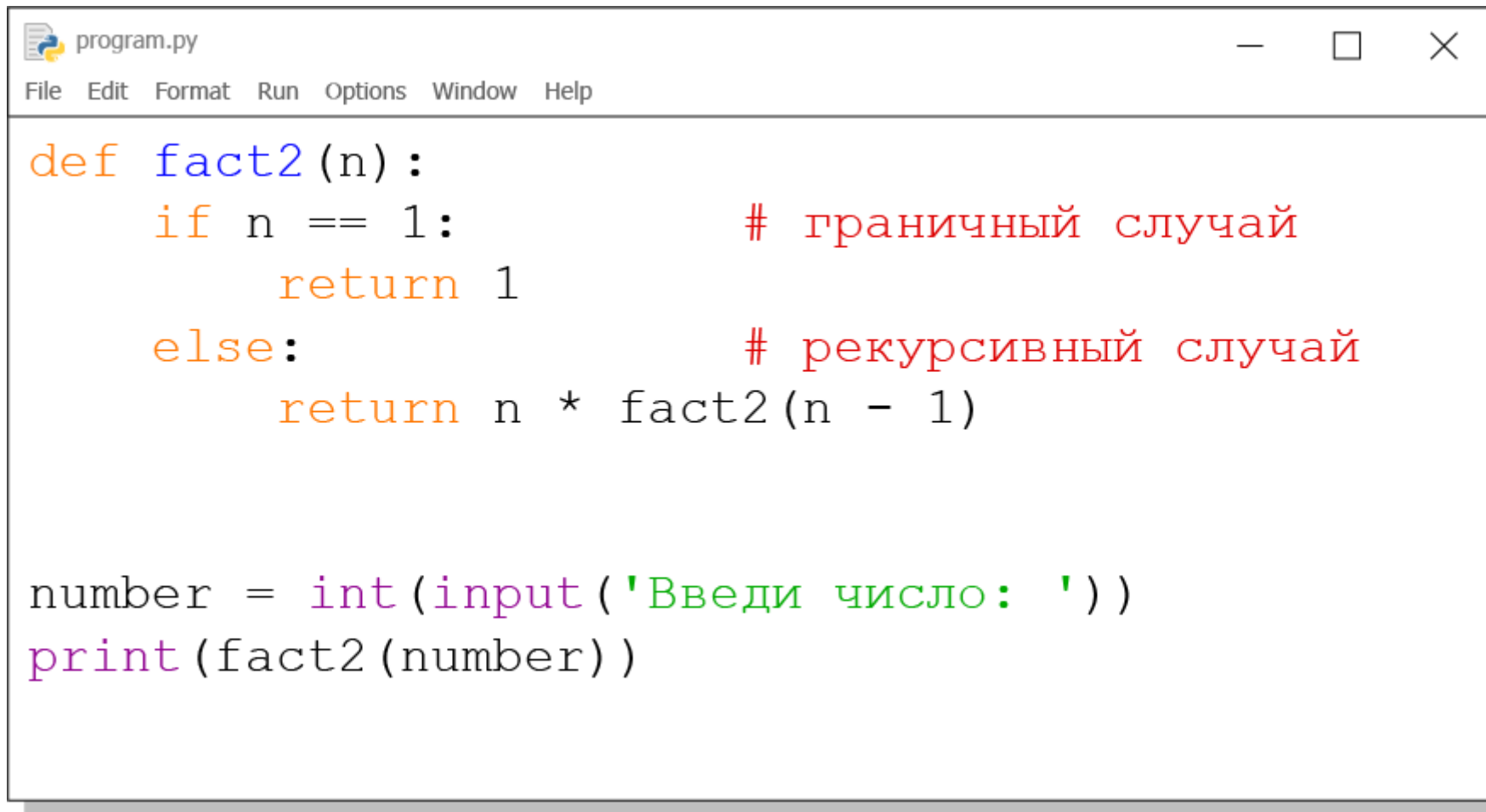
Очевидно, что в примере 1 мы повторяем одно и тоже действие – такой прием называют итерацией, а в примере 2 мы вновь и вновь обращаемся к функции нахождения факториала – такой прием называют рекурсией.

Если переменная  $n$  в нашем примере равна 5, то пошагово наша программа с рекурсивной функцией будет выполняться так:  $5! = 5 * 4!$ ,  $4! = 4 * 3!$  и так далее до граничного случая  $1! = 1$ . То есть каждый последующий факториал включает в себя определение предыдущего.

Другими словами, рекурсивная функция на каждом шаге вызывает саму себя. Особое внимание обратим на граничный случай, при котором функция завершает работу и возвращает данные в основную программу. Неправильное оформления выхода из рекурсии может привести к бесконечной цепочке действия. Поэтому очень важно при разработке рекурсивной



функции прежде всего подумать об условии завершения рекурсии. В общем виде рекурсивную функцию можно представить как две части:



```
def fact2(n):  
    if n == 1:                # граничный случай  
        return 1  
    else:                    # рекурсивный случай  
        return n * fact2(n - 1)  
  
number = int(input('Введи число: '))  
print(fact2(number))
```

*Рекурсия в Python имеет ограничение в 3000 слоев.*