



### Методические рекомендации по теме

«Сложные условия с помощью **elif**»

#### Цель:

- знакомство со сложными условиями на языке Python, решение программных прикладных задач с использованием оператора **elif**.

#### Задачи:

- расширение кругозора обучающихся в области информатики и программирования;
- объяснение логики применения сложного условия **elif**;
- решение программных прикладных задач, приводящих к применению сложных условий;
- ранняя профориентация школьников, профессиональная деятельность программиста;
- развитие интеллектуальных способностей, логического и критического мышления.

#### Планируемые результаты

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

*Предметные:* обучающиеся получают представления: о программировании сложных условий на языке Python; об использовании оператора **elif** в языке программирования Python; о прикладном использовании сложных условий в программных проектах; о возможностях и особенностях применения сложных условий в практике работы программиста.

*Метапредметные:* обучающиеся получат возможность владения обще предметными понятиями «иначе»; владения информационно-логическими умениями; умениями самостоятельно планировать пути достижения целей; умениями принятия решений и осуществления осознанного выбора; повысят уровень ИКТ-компетентности.

## **Материалы к занятию**

Приложение 1: Сценарный план ролика

Приложение 2: Домашнее задание и практика

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

Приложение 4: Диалоговая отладка программ: пошаговое выполнение, просмотр значений величин, отладочный вывод, выбор точки останова (дополнительно).

## **Ход проведения урока**

### **1. Организационный момент.**

#### **Мотивация на учебную деятельность.**

Приветствие учащихся, сообщение темы и целей занятия (мы узнаем, что такое сложное условие в программировании и как работать с такими условиями на языке Python; мы научимся использовать условные операторы **if** и **else** вместе с новым оператором **elif**, который позволяет создавать алгоритмы без ограничения на количество выполняемых условий. сделаем несколько программных проектов, чтобы понять как сложные операторы используются программистами, убедимся в необходимости тестирования кода).

#### **Проблемная дискуссия** по вопросам:

- Чем на ваш взгляд сложное условие отличается от простого (приведите примеры)?
- Какими словами можно описать сложное условие?

- Зачем нужно сложное условие в программировании?

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

- в реальной жизни мы часто встречаемся с ситуациями, где требуется сделать выбор не из двух вариантов, а из большого количества;
- использование сложных условий позволяет программисту реализовать в коде неограниченные возможности выбора вариантов;
- условие можно описать словом (**если, то, иначе**).

Преподаватель называет ученикам тему и цели урока.

## **2. Вводный блок.**

### **Тема.**

Преподаватель при необходимости останавливая трансляцию, комментируя дополнительно тему занятия.

*\*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.**

### Сложные условия с помощью **elif**.

Продолжение демонстрации ролика с дальнейшим обсуждением вопросов:

- Для чего нужен оператор **elif**?
- Какой синтаксис используется в Python для данного оператора?
- В чем отличие оператора **elif** от оператора **if**?
- В каком порядке нужно располагать операторы условия **if**, **elif**, **else** в коде?

*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. сцены 8 – 14*

### 5. Блок заданий.

**Проекты: «Автомат с напитками», «Средний балл».**

К началу демонстрации блока заданий ученики должны занять рабочие места и запустить Python (терминал IDLE) на своих компьютерах.

**«Автомат с напитками»:** включает *практическое задание 1* с таймером; после завершения работы таймера демонстрируется разбор задания. Задание представляет собой этапы создания программного проекта с использованием условий.

**После выполнения задания** ученики получают работающий продукт – программу выводящую на экран информацию в зависимости от выбора пользователя.

**«Средний балл»:** включает *практическое задание 2* с таймером; после завершения работы таймера демонстрируется разбор задания. Задание представляет этапы создания проекта с использованием сложного условия и переменной **score**.

**После выполнения задания** ученики получают работающий продукт – программу для определения характеристики ученика по его среднему баллу.

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

*\*см. сцены 15 – 20*

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

Завершаем демонстрацией ролика и кратким обобщением материалов занятия. Преподаватель возвращается к зафиксированному 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. сцену 21*

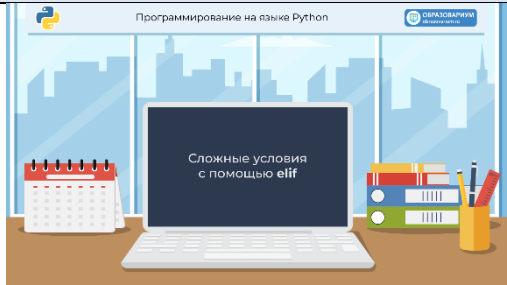
## Сценарный план видеоролика

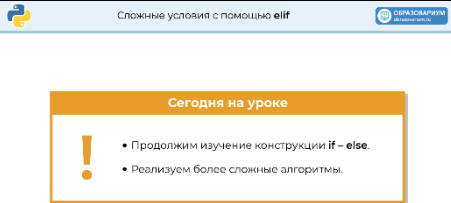
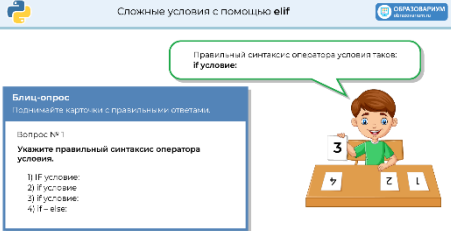
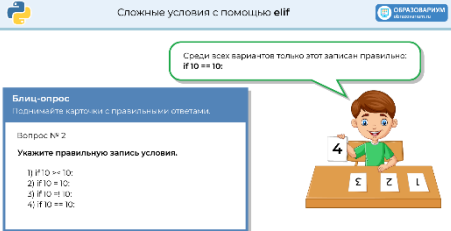
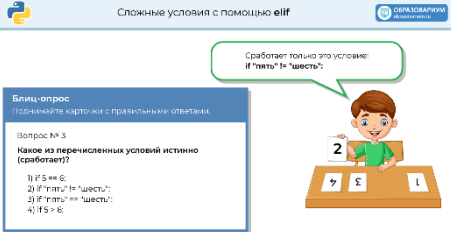
В таблице «Содержание видеоролика» представлены:

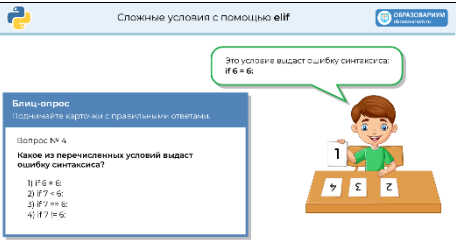
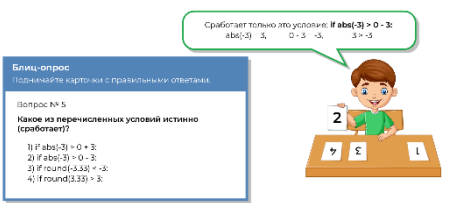

- название блоков видеоролика (тайминг);
- краткое описание содержания в каждом блоке;
- фрагменты из видеоролика, относящиеся к соответствующему блоку;
- номера сцен в каждом блоке.

*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.  Сложные условия с помощью <b>elif</b>	 <p>Сцена 1</p>	1 2

	<p>При необходимости, останавливаем трансляцию и дополнительно комментируем.</p> <p>Мы продолжим знакомство с условным оператором, представленным конструкцией <b>elif</b> и познакомимся с командой для организации более сложных алгоритмов.</p>		
<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 правильный синтаксис оператора условия.</p> <ol style="list-style-type: none"> <li>1) if условие:</li> <li>2) if условие</li> <li>3) if условие:</li> <li>4) if – else:</li> </ol>	<p>Сцена 2</p> 	<p>3</p> <p>4</p> <p>5</p> <p>6</p> <p>7</p>
	<p><b>Второй вопрос.</b> Укажите правильную запись условия.</p> <ol style="list-style-type: none"> <li>1) if 10 &gt;&lt; 10:</li> <li>2) if 10 = 10:</li> <li>3) if 10 != 10:</li> <li>4) if 10 == 10:</li> </ol>	<p>Сцена 3</p> 	
	<p><b>Третий вопрос.</b> Какое из перечисленных условий истинно (сработает)?</p> <ol style="list-style-type: none"> <li>1) if 5 == 6:</li> <li>2) if «пять» != «шесть»:</li> <li>3) if «пять» == «шесть»:</li> <li>4) if 5 &gt; 6:</li> </ol>	<p>Сцена 4</p> 	<p>Сцена 5</p>

	<p><b>Четвертый вопрос.</b> Какое из перечисленных условий выдаст ошибку синтаксиса?</p> <ol style="list-style-type: none"> <li>1) if 6 = 6:</li> <li>2) if 7 &lt; 6:</li> <li>3) if 7 == 6:</li> <li>4) if 7 != 6:</li> </ol>	 <p>Сцена 6</p>	
	<p><b>Пятый вопрос.</b> Какое из перечисленных условий истинно (сработает)?</p> <ol style="list-style-type: none"> <li>1) if abs(-3) &gt; 0 + 3:</li> <li>2) if abs(-3) &gt; 0 - 3:</li> <li>3) if round(-3.33) &lt; -3:</li> <li>4) if round(3.33) &gt; 3:</li> </ol>	 <p>Сцена 7</p>	
<p>Теоретический блок.</p> <p><b>Сложные условия с помощью elif</b></p>	<p><i>При необходимости преподаватель может поставить ролик на паузу и дать дополнительные пояснения по материалу</i></p> <p>Перейдем к изучению новой темы.</p> <p>Оператор условия <b>if-else</b> имеет два варианта развития событий, первый – при выполнении условия и второй – во всех остальных случаях.</p> <p>Рассмотрим ситуацию, когда <b>двух вариантов недостаточно</b>.</p>	 <p>Сцена 8</p>	<p>8</p> <p>9</p> <p>10</p> <p>11</p> <p>12</p> <p>13</p> <p>14</p>



Представим ситуацию. Нам предложены суп, котлета или мороженое. И соответственно, столовые приборы – столовая ложка, вилка, десертная ложка.

Мы можем составить пары, используя только оператор **if**.

Однако, такой алгоритм (по машинным меркам) будет работать очень долго, трижды проверяя условие на истинность. Как изменить этот алгоритм?

Рассмотрим второй вариант.

Здесь первое и третье условие созданы при помощи уже знакомых нам операторов **если – иначе**

А вот второе прописано при помощи двух слов – **иначе если**.

Это еще одна команда оператора условия, и она называется **elif**.

Первые две буквы она заимствует у оператора **else**, а две последние – у оператора **if**.

Как же работает эта команда и какой у нее синтаксис?

Начнем с того, что команд **elif** в условном операторе может быть сколько угодно. Главное – они должны располагаться после команды **if**, которая (как и **else** – всегда одна). И еще - **elif** всегда требует себе условие.



Сцена 9

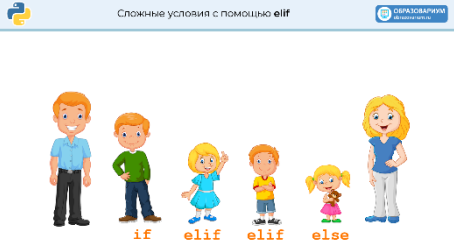
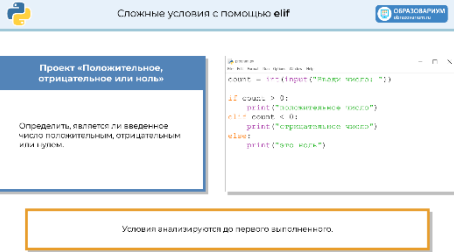
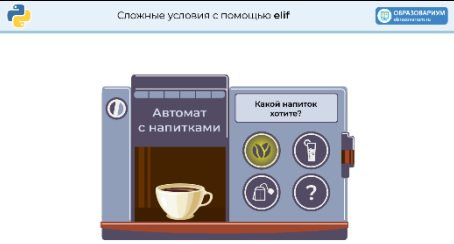


Вариант 1	Вариант 2
<p>Если суп – столовая ложка. Если котлета – вилка. Если мороженое – десертная ложка.</p> <p>Три оператора <b>if</b>.</p>	<p>Если суп – столовая ложка. Иначе если котлета – вилка. Иначе – десертная ложка.</p> <p><b>elif</b> – иначе если <b>if</b> – <b>elif</b> – <b>else</b></p>

Сцена 10

Условный оператор	
<b>if</b>	<ul style="list-style-type: none"> <li>• обязательно должен быть</li> <li>• всегда первый</li> <li>• всегда один</li> <li>• требует условие</li> </ul>
<b>elif</b>	<ul style="list-style-type: none"> <li>• необязательно должен быть</li> <li>• количество неограниченно</li> <li>• всегда после <b>if</b></li> <li>• требует условие</li> </ul>
<b>else</b>	<ul style="list-style-type: none"> <li>• необязательно должен быть</li> <li>• всегда последний</li> <li>• всегда один</li> <li>• не требует условия</li> </ul>

Сцена 11

	<p>Схему <b>if-elif-else</b> упрощенно можно сравнить с семьей, где есть дети.</p> <p>Если ребенок один – то он всегда старший. Это - <b>if</b></p> <p>Если их двое – то старший это <b>if</b>, младший – <b>else</b>.</p> <p>А если детей больше двух, то старший по-прежнему <b>if</b>, младший по-прежнему <b>else</b>, а вот все остальные – это <b>elif</b>.</p>	 <p>Сцена 12</p>	
	<p>Рассмотрим, как будет выглядеть программа с прошлого урока, которая определяет является ли введенное число положительным, отрицательным или нулем. Как видим, первое условие записано через <b>if</b>, второе через <b>elif</b>, а <b>else</b> условия не требует.</p> <p>Такая программа выполнится гораздо быстрее, так как будут анализироваться условия только до первого выполненного. И после этого дальше даже не будут рассматриваться варианты. То есть, если обнаружится что число положительное – варианты с <b>elif</b> и <b>else</b> просто игнорируются, так как выбор уже сделан.</p>	 <p>Сцена 13</p>	
	<p>Теперь создадим свой проект под названием «<b>Автомат с напитками</b>»</p> <p>Программа должна спросить пользователя: какой напиток хотите?</p> <p>Если будет введено <b>кофе</b>, то напечатается кофе сварен, если <b>чай</b> – то чай заварен, если <b>сок</b> – сок налит.</p> <p>На все остальные запросы будет ответ – нет в наличии</p>	 <p>Сцена 14</p>	

Блок заданий.  
Практические  
задания:  
Задание 1  
Задание 2

После окончания дикторского текста запускается таймер на 7 минут.

### Задание 1. Проект «Автомат с напитками»

Алгоритм проекта перед вами.

При создании проекта нам понадобилась переменная для хранения ответа.

Ее обозначили **answer**, что переводится как слово «ответ».

Текст вопроса желательно сделать внутри функции **input**.

Не забудьте, что в проекте вводится и сравнивается **текстовая** информация!

### Разбор задания 1.

Код нашего проекта может выглядеть вот так.

Поскольку у нас четыре условия: кофе, чай, сок и другой напиток, то мы используем команду **if**, две команды **elif** и **else**.

Для написания равенства используется знак двойное равно, после условия 4 отступа.


На будущее – прежде чем писать код, необходимо посчитать сколько всего условий в алгоритме, чтобы понимать какие команды использовать.

Как видим – с увеличением условий количество команд **elif** также растет.

Сложные условия с помощью **elif**

Алгоритм «Автомат с напитками»

- Вывести фразу «Какой напиток?»
- Если длина строки ответа не равна пустой строке, то вывести **answer**
- Если **answer** равен «кофе»  
Напечатать «кофе сварен»  
Иначе если **answer** равен «чай»  
Напечатать «чай заварен»  
Иначе если **answer** равен «сок»  
Напечатать «сок налит»  
Иначе  
Напечатать «нет в наличии»



Сцена 155

Сложные условия с помощью **elif**

Алгоритм «Автомат с напитками»

- Вывести фразу «Какой напиток?»
- Если длина строки ответа не равна пустой строке, то вывести **answer**
- Если **answer** равен «кофе»  
Напечатать «кофе сварен»  
Иначе если **answer** равен «чай»  
Напечатать «чай заварен»  
Иначе если **answer** равен «сок»  
Напечатать «сок налит»  
Иначе  
Напечатать «нет в наличии»

```
if __name__ == '__main__':
    answer = input("Какой напиток? ")

    if answer == "кофе":
        print("кофе сварен")
    elif answer == "чай":
        print("чай заварен")
    elif answer == "сок":
        print("сок налит")
    else:
        print("нет в наличии")
```

Сцена 166

Сложные условия с помощью **elif**

Количество условий	Конструкция
1 условие	<b>if</b>
2 условия	<b>if - else</b>
3 условия	<b>if - elif - else</b>
4 условия	<b>if - elif - elif - else</b>
5 условий	<b>if - elif - elif - elif - else</b>

Сцена 17

15

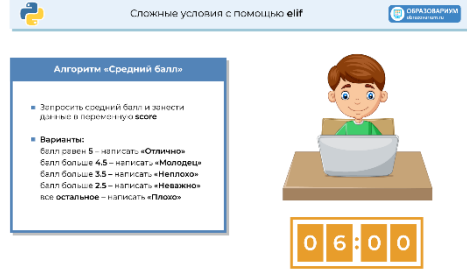
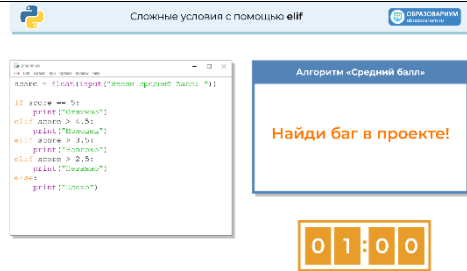
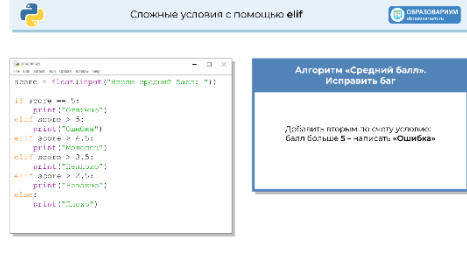
16

17

18

19

20

	<p><i>После окончания дикторского текста запускается таймер на 6 минут.</i></p> <p><b>Задание 2. Проект «Средний балл»</b>          Программа запросит средний балл ученика и выдаст свою характеристику.</p> <p><b>Варианты:</b> балл равен 5 – написать «Отлично»; балл больше 4.5 – написать «Молодец»; балл больше 3.5 – написать «Неплохо»; балл больше 2.5 – написать «Неважно»; все остальное – написать «Плохо».</p> <p>Также нам понадобится числовая дробная переменная <b>score</b>.</p>	 <p>Сцена 18</p>	
	<p><b>Разбор задания 2.</b></p> <p>Ваш код может выглядеть примерно так.</p> <p>В нем имеется дробная переменная <b>score</b>, команда <b>if</b>, три команды <b>elif</b> и команда <b>else</b>.</p> <p>Запустите несколько наш код, вводя различные числа и попробуйте найти в проекте баг.</p> <p><i>Дается минута на поиск бага.</i></p>	 <p>Сцена 19</p>	
	<p><b>Разбор задания 2 (продолжение).</b></p> <p>Баг в том, что если ввести средний балл больше, чем пять, то будет напечатано «Молодец».</p> <p>Потому что все подобные числа подпадают под действие второго условия: если <b>score</b> больше четырех с половиной.</p> <p>Как это исправить?</p> <p>Необходимо вторым условием (после <b>score</b> равно 5) создать еще одно: балл больше 5 – вывести «Ошибка»</p>	 <p>Сцена 20</p>	

<p>Блок завершения занятия.</p> <p><b>Рефлексия. Сообщение домашнего задания</b></p>	<p><i>Завершаем демонстрацией ролика и кратким обобщением материалов занятия.</i></p> <p><b>Подведем итоги:</b></p> <ul style="list-style-type: none"> <li>- узнали, что, расширив конструкцию условного оператора с помощью команды <b>elif</b>, можно создавать алгоритмы без ограничения на количество поставленных условий;</li> <li>- количество условий влияет на набор команд, которые нам необходимы для создания алгоритма.</li> </ul> <p><i>Преподаватель дает ученикам домашнее задание к следующему занятию (Приложение 2).</i></p>	<div data-bbox="1509 228 1962 483"> </div> <p>Сцена 21</p>	<p>21</p>
--	---	--	-----------

### Домашнее задание

Придумать и записать свой алгоритм, в котором будет несколько различных условий.

### Практика

Проект «Рекомендации»

Придумайте и составьте рекомендации по просмотру фильмов и телепрограмм для нескольких возрастных групп.

Запросите возраст пользователя. Выведите рекомендации с учетом его возрастной группы.

Проект «Рост и вес»

Запросите рост и вес пользователя.

Рассчитайте идеальный вес по формуле Лоренца и индекс массы тела (ИМТ). Выведите результаты расчетов и рекомендации по весу.

*\*Для расчетов ИМТ и идеального веса воспользуйтесь поиском в сети Интернет.*

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

«Сложные условия с помощью **elif**»

**В начале занятия** можно вспомнить все виды алгоритмов, и в чем их принципиальное отличие друг от друга. Пусть ребята также приведут их примеры. Повторите правила и синтаксис применения команд **if-else**, и о том, с помощью каких символов можно записать необходимые нам условия (больше, меньше, равно, не равно). Для большей наглядности можно записать их на доске.

**Перед просмотром** блока повторения из ролика необходимо раздать дидактический материал для выполнения заданий из блока повторение (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 теоретическим блоком** можно еще раз уточнить, что представляет собой условный алгоритм (варианты событий при наличии условий) и попросить привести примеры условных алгоритмов из жизни.

**После теоретического блока** можно остановить ролик и попросить привести примеры, где встречается минимум три условия (вариант – сигналы светофора, времена года). Попросите сформулировать данные алгоритмы, используя слова «если», «иначе если», «иначе».

**Перед блоком заданий** необходимо проследить, чтобы у всех был открыт компьютерный терминал для практической работы.

**Если задание № 1** не вызовет сложностей, можно попросить прописать в алгоритм дополнительные условия (варианты: квас, вода, лимонад и собственно придуманные фразы). Обратите внимание, что вставлять эти условия необходимо после первого условия (с **if**), но перед последним (**else**).

**Задание № 2** выполняется в несколько этапов: основной код, поиск бага, дополнительное условие для исправления бага. Можно простимулировать тех ребят, кто быстро обнаружит баг в программе (его дает ввод числа больше 5).

**Обратите внимание**, что дополнительное условие надо прописать сразу за первым (с оператором **if**). Но если останется время – можно поэкспериментировать: попробовать разместить его в другом месте и посмотреть, что получится (другой баг).

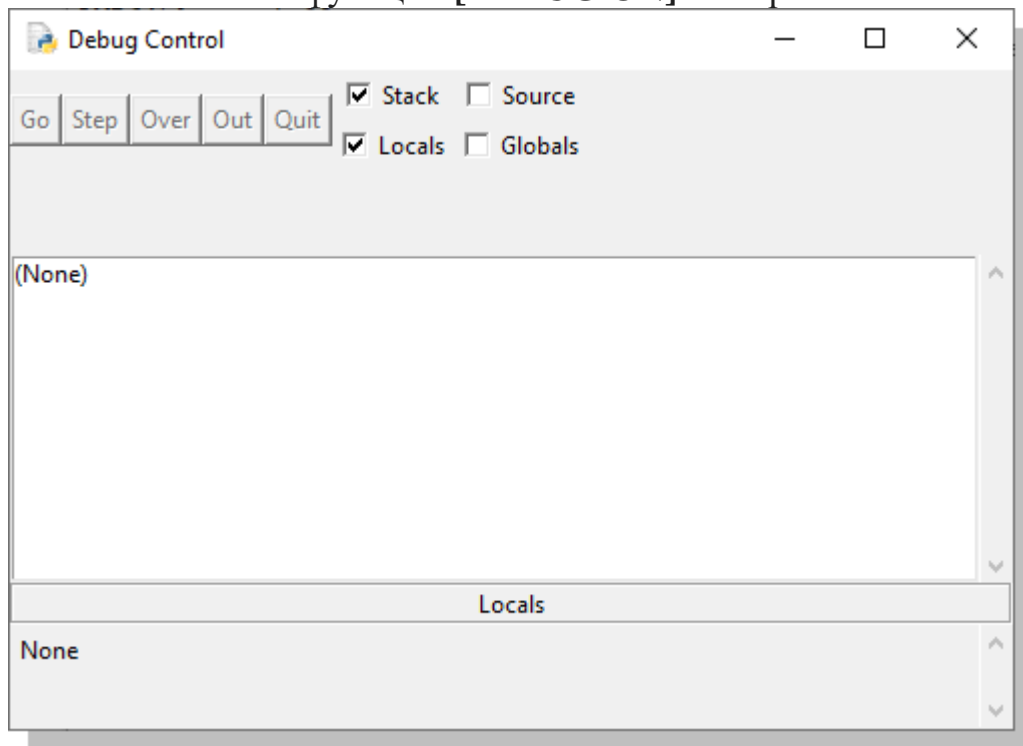
После каждого блока заданий, созданные проекты необходимо протестировать несколько раз с разными вводными данными.

Дополнительно

**Диалоговая отладка программ: пошаговое выполнение, просмотр значений величин, отладочный вывод, выбор точки останова.**

Python IDLE предоставляет инструменты для поиска и исправления ошибок в программном коде, то есть для отладки исходного кода программы.

Для запуска отладчика нужно в строке меню IDLE выбрать **Debug** → **Debugger**. В окне IDLE появится информация о включении этой функции **[DEBUG ON]** и откроется окно отладчика **Debug Control**.





В этом окне можно проверить значения переменных при выполнении кода, а также получить представления о том, как данные обрабатываются во время работы кода.

Для перемещения по коду в окне используются следующие кнопки:

**Go** – переход к следующей точке останова (**Breakpoint**).

**Step** – выполнить текущую строку и перейти к следующей.

**Over** – выполнить функцию и перейдите к следующей строке, но не делать паузу во время выполнения функции (если не существует точки останова).

**Out** – если текущая строка кода находится в функции, то нажмите эту кнопку, чтобы перейти к шагу **Out** этой функции. Другими словами, продолжайте выполнение этой функции, пока не вернетесь к ней.

*Будьте осторожны, потому что нет кнопки реверса! Вы можете только сделать шаг вперед во время выполнения вашей программы.*

В окне отладки также есть четыре флажка:

**Globals** – глобальная информация о программе.

**Locals** – локальная информация программы во время выполнения.

**Stack** – функции, которые запускаются во время выполнения.

**Source** – файл программы в редакторе IDLE.

В зависимости от выбора отражается соответствующая информация в окне отладки.

### **Точки останова (Breakpoint)**

**Breakpoint** – это строка кода, которую определяют как место, где интерпретатор должен приостановить выполнение кода (работает только при включенном режиме **DEBUG**).

Для установки точки останова (**Breakpoint**), на нужной строке кода щелкаем правой кнопкой мыши и в появившемся меню выбираем команду **Set Breakpoint**, отмена точки останова осуществляется аналогично командой **Clear**

**Breakpoint**. Выделение строки кода желтым маркером визуализирует индикацию установленной точки останова.

После установки точек останова и включения режима **DEBUG**, можно запускать программный код, как обычно.

Откроется окно отладчика, что позволит начать пошаговое выполнение кода вручную.