



Методические рекомендации по теме **«Пользовательские функции и методы работы с ними»**

Цель:

- дать представление о понятии и методах работы с функциями в языке Python.

Задачи:

- практика применения функций в Python;
- освоение основных методов работы с функциями 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.

Метапредметные: обучающиеся получают возможность владения общепредметными понятиями «функция», «метод», «управляющий цикл»; 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

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:

- Как бы вы объяснили понятие «функция»?
- Для чего могут быть использованы функции в программировании?
- Что такое «функция» в реальном мире?

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

- Функцией в Python называют служебную команду
- Также функцией называют часть кода, который написан отдельно от основной программы для решения отдельной задачи.

**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.

Пользовательская функция

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

Обсуждением вопросов по просмотренным материалам:

- Что такое пользовательская функция?
- Для чего пользовательские функции могут быть нужны на практике?
- На каком этапе разработки принято создавать пользовательские функции?

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 – 11, 26*

5. Блок заданий.

«Функции».

«Функции»: включает *блок практических заданий*, которые позволят провести прикладные разработки типовых задач для работы с функциями:

- Функция с двумя аргументами
- Функция без аргументов
- Получение данных из функции
- Возведение в квадрат

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

**см. сцены 12 – 25.*

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. сцена 27*

Приложение 1

Сценарный план видеоролика

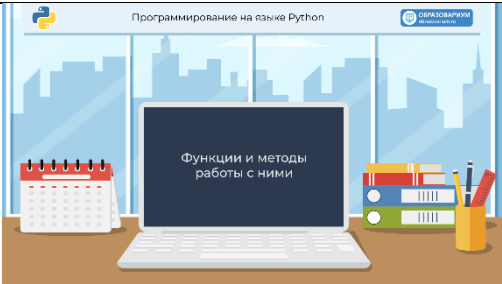

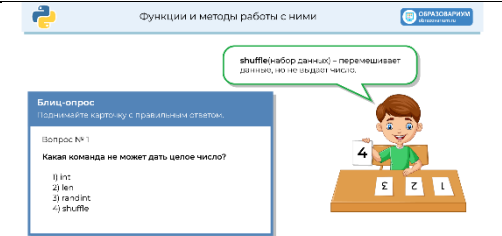
В таблице «Содержание видеоролика» представлены:

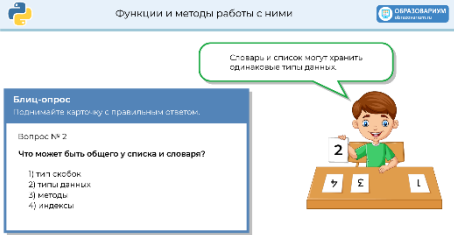
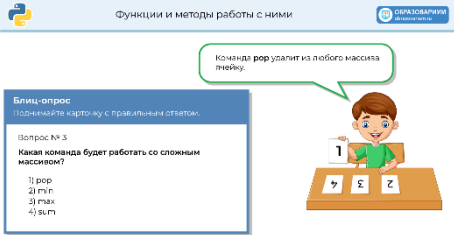
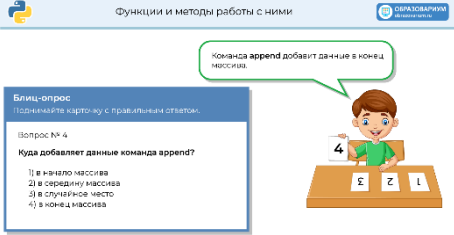
- название блоков видеоролика (тайминг);
- краткое описание содержания в каждом блоке;
- фрагменты из видеоролика, относящиеся к соответствующему блоку;
- номера сцен в каждом блоке.

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.

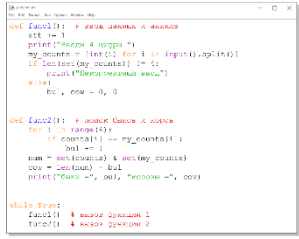
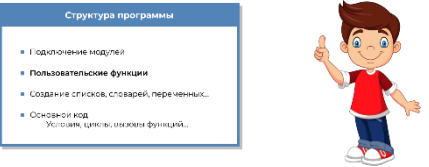
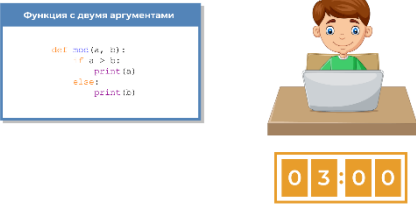
Таблица. Содержание видеоролика

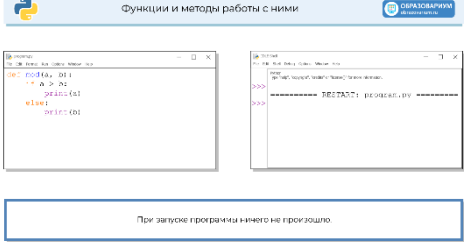
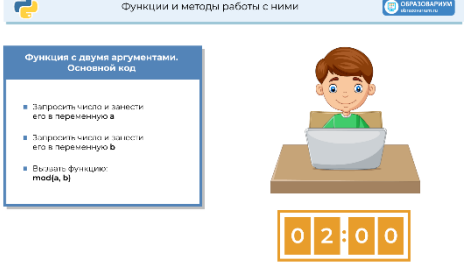
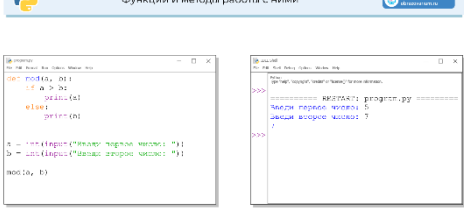
Название блока	Содержание блока и комментарии	Фрагменты из видеоролика	№ сцен
----------------	--------------------------------	--------------------------	--------

<p>Вводный блок. Мы узнаем</p>	<p><i>Обозначаем ученикам тему и цели урока.</i></p> <p>Пользовательские функции и методы работы с ними</p>	 <p>Сцена 1</p>	<p>1 2</p>
	<p>Мы начнем знакомство с таким понятием как функция. Этот термин уже неоднократно упоминался в наших уроках. Напомним, что под словом функция в Python подразумевается служебная команда, которая что-то делает с данными. Print, input, int, float, len. Все эти команды являются функциями и подсвечиваются в терминале одним и тем же цветом. Обозначим ещё одно значение функции. Функция – фрагмент программного кода, который написан отдельно от основной программы для решения конкретной задачи.</p>	 <p>Сцена 2</p>	
<p>Блок повторения. Блиц-опрос</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>Первый вопрос. Какая команда не может дать целое число?</p> <ol style="list-style-type: none"> 1) int 2) len 3) randint 4) shuffle <p><i>Ответ 4. shuffle (набор данных) – перемешивает данные, но не выдает число.</i></p>	 <p>Сцена 3</p>	<p>3 4 5 6 7</p>

	<p>Второй вопрос. Что может быть общего у списка и словаря?</p> <ol style="list-style-type: none"> 1) Тип скобок 2) Типы данных 3) Методы 4) Индексы <p><i>Ответ 2. Словарь и список могут хранить одинаковые типы данных.</i></p>	 <p>Функции и методы работы с ними</p> <p>Сценарий 4</p> <p>Блиц-опрос Поднимайте карту, жу с правильным ответом.</p> <p>Вопрос № 2 Что может быть общего у списка и словаря?</p> <ol style="list-style-type: none"> 1) тип скобок 2) типы данных 3) методы 4) индексы <p>Словари и списки могут хранить одинаковые типы данных.</p>	
	<p>Третий вопрос. Какая команда будет работать со сложным массивом?</p> <ol style="list-style-type: none"> 1) pop 2) min 3) max 4) sum <p><i>Ответ 1. pop удалит из любого массива ячейку.</i></p>	 <p>Функции и методы работы с ними</p> <p>Сценарий 5</p> <p>Блиц-опрос Поднимайте карту, жу с правильным ответом.</p> <p>Вопрос № 3 Какая команда будет работать со сложным массивом?</p> <ol style="list-style-type: none"> 1) pop 2) min 3) max 4) sum <p>Команда pop удалит из любого массива ячейку.</p>	
	<p>Четвертый вопрос. Куда добавляет данные команда append?</p> <ol style="list-style-type: none"> 1) В начало массива 2) В середину массива 3) В случайное место 4) В конец массива <p><i>Ответ 4. append добавит данные в конец массива.</i></p>	 <p>Функции и методы работы с ними</p> <p>Сценарий 6</p> <p>Блиц-опрос Поднимайте карту, жу с правильным ответом.</p> <p>Вопрос № 4 Куда добавляет данные команда append?</p> <ol style="list-style-type: none"> 1) в начало массива 2) в середину массива 3) в случайное место 4) в конец массива <p>Команда append добавит данные в конец массива.</p>	

	<p>Пятый вопрос. Какие индексы у числа «8»?</p> <p>[[1, 1], [2, 2], [11, 10, 9, 8, 7]]</p> <p>Поднимите карточки с соответствующими числами.</p> <p><i>Ответ 2 и 3. Индексы числа 8 – [2][3]</i></p> <p>[[1, 1], [2, 2], [11, 10, 9, 8, 7]]</p> <p>0 1 2 3 4</p> <p>0 1 2</p>	<p>Функции и методы работы с ними</p> <p>Блиц-опрос</p> <p>Поднимайте карточки с правильными ответами.</p> <p>Вопрос №5</p> <p>Какие индексы у числа «8»?</p> <p>[[1, 1], [2, 2], [11, 10, 9, 8, 7]]</p> <p>Поднимите карточки с соответствующими числами.</p> <p>Индексы числа 8 – [2][3]</p> <p>Сцена 7</p>	
<p>Теоретический блок.</p> <p>Пользовательская функция</p>	<p>Мы познакомимся с так называемой пользовательской функцией. Это часть программы, которая отдельно создается пользователем. Для чего это нужно?</p> <p>Представьте, что есть код, который должен выполняться несколько раз. Этот код можно поместить внутри цикла, но помимо него, там еще наверняка будут и другие команды. Тело цикла станет очень большим и трудным для понимания.</p>	<p>Функции и методы работы с ними</p> <p>Зачем нужны пользовательские функции</p> <ul style="list-style-type: none"> ■ Выполнение кода несколько раз ■ Выполнение кода в разных местах программы ■ Улучшение структуры кода ■ Облегчение понимания кода <p>Сцена 8</p>	<p>8</p> <p>9</p> <p>10</p> <p>11</p>
	<p>Давайте рассмотрим фрагмент кода нашей программы «Быки и коровы».</p> <p>Внутри цикла находятся три раздела: ввод чисел игроком, поиск быков, поиск коров</p> <p>Код идет строка за строкой и только комментарии позволяют нам понять, какой фрагмент за что отвечает.</p>	<p>Функции и методы работы с ними</p> <pre> def is_valid(): while True: print("Введите 4 цифры ") my_number = [int(i) for i in input().split()] if len(my_number) != 4: print("Некорректная длина") else: bulls, cows = 0, 0 for i in range(4): if my_number[i] == my_secret[i]: bulls += 1 for i in range(4): if my_number[i] in my_secret: cows += 1 print(f"Быки: {bulls}, Коровы: {cows}") </pre> <p>Сцена 9</p>	

	<p>Этого можно избежать, сделав две пользовательские функции – одна принимает данные, вторая ищет «быков и коров».</p> <p>Эти функции мы бы расположили отдельно и от цикла и друг от друга. А чтобы код работал – в теле цикла оставили бы только вызов этих функций.</p> <p>Теперь внутри игрового цикла всего две команды, а сам код разбит на части и стал более понятен.</p>	 <p>Сцена 10</p>	
	<p>Это можно сравнить с тем, что у вас на жестком диске установлена игра, а на рабочем столе имеется ярлык для ее запуска. То есть мы не храним всю информацию в одном месте, а распределяем ее как нам удобно. Так ваш рабочий стол можно представить в виде основного цикла программы: в нем только ярлыки, а сама информация – в других местах</p> <p>Теперь, когда мы знаем зачем создавать пользовательскую функцию, давайте посмотрим, как и где это делать. Пользовательская функция создается всегда в начале программы, после подключения модулей (если они есть)</p>	 <p>Сцена 11</p>	
<p>Блок заданий.</p> <p>Практические задания:</p> <p>Задание 1</p>	<p><i>После окончания дикторского текста запускается таймер на 5 мин.</i></p> <p>Задание 1. «Функция с двумя аргументами». Начало</p> <pre>def mod(a, b): if a > b: print(a) else: print(b)</pre> <p><i>Функция создается при помощи служебной команды def, потом указывается имя, которое вы ей дадите, круглые скобки (они могут быть пустыми, или содержать имена переменных) и в конце двоеточие. Наша функция по имени mod будет содержать внутри скобок две переменные (они называются аргументы)</i></p>	 <p>Сцена 12</p>	<p>12 13 14 15</p>

	<p>Разбор задания 1. Наша функция должна получить данные, занести их в переменные a и b, сравнить их и напечатать одно из чисел. Почему должна? Потому что если запустить программу, то ничего не произойдет. Наша функция как бы находится в засаде и ждет своего вызова. Давайте продолжим код</p>	 <p>Сцена 13</p>	
	<p><i>После окончания дикторского текста запускается таймер на 2 мин.</i></p> <p>Задание 1. Алгоритм «Функция с двумя аргументами. Основной код</p> <ul style="list-style-type: none"> ■ Запросить число и занести его в переменную a ■ Запросить число и занести его в переменную b ■ Вызвать функцию: ■ mod(a, b) 	 <p>Сцена 14</p>	
	<p>Разбор задания 1. Ваш код может выглядеть так.</p> <pre>def mod(a, b): if a > b: print(a) else: print(b) a = int(input("Введи первое число: ")) b = int(input("Введи второе число: ")) mod(a, b)</pre>	 <p>Сцена 15</p>	

Блок заданий.
Практические задания:
Задание 2

После окончания дикторского текста запускается таймер на 3 мин.

Задание 2. Алгоритм «Функция без аргументов»

```
def mod():
    Запросить число и занести его в переменную a
    Запросить число и занести его в переменную b
    if a > b:
        print (a)
    else:
        print (b)
```

Разбор задание 3. Код программы будет выглядеть так:

```
def mod():
    a = int(input("Введи первое число: "))
    b = int(input("Введи второе число: "))
    if a > b:
        print(a)
    else:
        print(b)
```

В чем разница между программами? Обе запрашивают числа и выводят то, которое больше. Но во втором проекте – весь код, за исключением вызова, находится внутри функции. При этом есть одна особенность. Попробуйте запросить через терминал чему равна переменная **a**? Получим сообщение об ошибке: такое имя программе не известно. Почему?

Данные пользовательской функции для основной программы недоступны.

Это можно сравнить с вашим телефоном, который перестал принимать звонки, как только вы уехали в другую страну и не подключили роуминг. Пользовательская функция – это фактически другая страна по отношению к основному коду. Поэтому если нам нужно извлечь данные из функции, то надо воспользоваться своеобразным роумингом.

Двумерные массивы и генераторы

Алгоритм «Вычисление суммы массива»

(предложеное решение)

- `summa = 0`
- Организовать цикл до 3
- `summa += sum(data[i])`
- Напечатать `summa`

Сцена 16

Функции и методы работы с ними


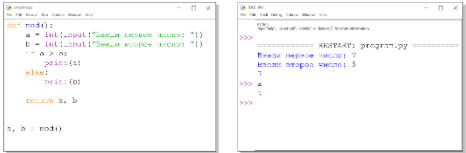

Сцена 17

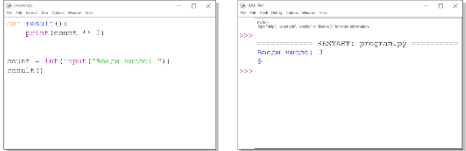
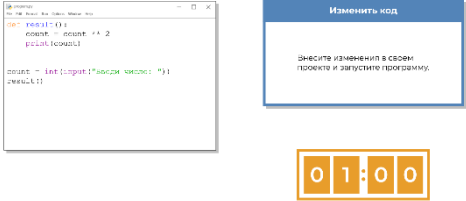
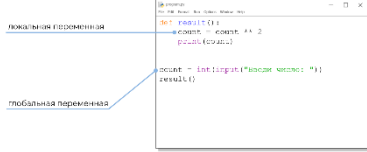
Функции и методы работы с ними

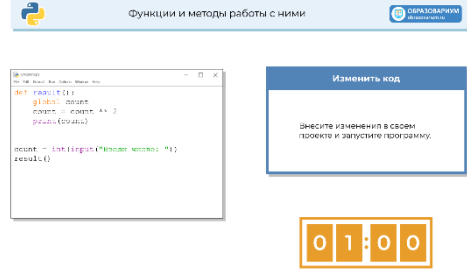
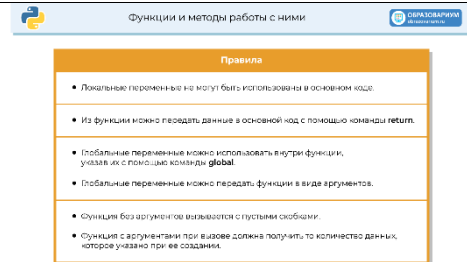
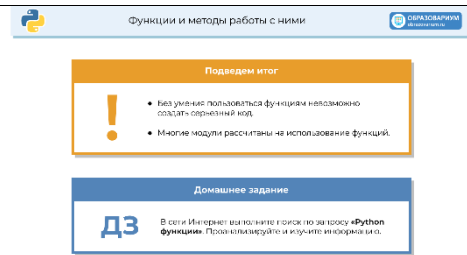
Данные пользовательской функции для основной программы недоступны.

Сцена 18

15
16
17

<p>Блок заданий. Практические задания: Задание 3</p>	<p><i>После окончания дикторского текста запускается таймер на 1 мин.</i></p> <p>Задание 3. Алгоритм «Получение данных из функции»</p> <ul style="list-style-type: none"> Допишите последней строкой функции: return a, b Измените вызов функции на: a, b = mod() 	<p>Функции и методы работы с ними</p> <p>Получение данных из функции</p> <ul style="list-style-type: none"> Дополнить последней строкой функции: <code>return a, b</code> Использовать функцию на: <code>a, b = mod()</code>  <p>Сцена 19</p>	<p>19 20</p>
	<p>Разбор задание 3. Код программы будет выглядеть так:</p> <pre>def mod(): a = int(input("Введи первое число: ")) b = int(input("Введи второе число: ")) if a > b: print(a) else: print(b) return a, b a, b = mod()</pre>	<p>Функции и методы работы с ними</p>  <p>Сцена 20</p>	
<p>Блок заданий. Практические задания: Задание 4</p>	<p><i>После окончания дикторского текста запускается таймер на 3 мин.</i></p> <p>Задание 4. Алгоритм «Возведение в квадрат»</p> <ul style="list-style-type: none"> def result (): <ul style="list-style-type: none"> Напечатайте квадрат числа count Запросить число и занести его в count result () 	<p>Функции и методы работы с ними</p> <p>Возведение в квадрат</p> <ul style="list-style-type: none"> def result(): <ul style="list-style-type: none"> Напечатать квадрат числа count Запросить число и занести его в count result()  <p>Сцена 21</p>	<p>21 22 23 24 25</p>

	<p>Разбор задание 4. Код программы будет выглядеть так:</p> <pre>def result(): print(count ** 2) count = int(input("Введи число: ")) result()</pre> <p>Можем ли мы сделать вывод, что переменные, которые созданы в основном коде спокойно работают внутри функции?</p>	 <p>Сцена 22</p>	
	<p>Нет, не совсем так.</p> <p>Если мы попробуем для начала внутри функции изменить нашу переменную, а потом вывести ее на экран, то получим сообщение об ошибке.</p> <p>Внесите изменения в своем проекте и запустите программу.</p> <p><i>После окончания дикторского текста запускается таймер на 1 мин.</i></p>	 <p>Сцена 23</p>	
	<p>Что произошло?</p> <p>Дело в том, что переменные, которые создаются и используются в основном коде называются глобальными, а те, которые создаются или используются в вычислениях внутри функций – называются локальными.</p> <p>Python воспринимает их не как одну переменную, а как разные.</p> <p>Как братьев-близнецов: выглядят одинаково, но это разные люди.</p>	 <p>Сцена 24</p>	

	<p>А можно это изменить? Да, добавьте в начало функции команду global с именем переменной.</p> <p>Теперь программа даст разрешение переменной count работать внутри функции.</p> <p>Внесите изменения в своем проекте и запустите программу.</p> <p><i>После окончания дикторского текста запускается таймер на 1 мин.</i></p>	 <p>Сцена 25</p>	
Теоретический блок	<p>Давайте сформулируем некоторые правила:</p> <ul style="list-style-type: none"> - переменные, которые созданы внутри функции (локальные) не могут быть использованы в основном коде, но могут передать свои данные при помощи команды return. - переменные, которые созданы внутри основного кода (глобальные) всегда могут быть использованы внутри функции при помощи команды global, а еще могут передать свои данные в виде аргументов - если функция создана с пустыми скобками (без аргументов), то и вызывается она с пустыми скобками. - если функция создана с аргументами, то она должна при вызове получить то количество данных, которое указано при её создании. 	 <p>Сцена 26</p>	26
Блок завершения занятия. Рефлексия. Сообщение домашнего задания	<p><i>Завершаем демонстрацией ролика и кратким обобщением материалов занятия.</i></p> <p>Подведем итоги.</p> <p>Мы узнали:</p> <ul style="list-style-type: none"> ■ Без умения пользоваться функциям невозможно создать серьезный код. ■ Многие модули рассчитаны на использование функций. <p><i>Преподаватель дает ученикам домашнее задание к следующему занятию (Приложение 2).</i></p>	 <p>Сцена 27</p>	27

Домашнее задание

Найдите в различных источниках (интернет, литература) информацию по использованию функций 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 «Два рандома»

Напишите три функции.

Первая функция запрашивает у пользователя и возвращает 2 числа: минимальное значение и максимальное. Аргументы в функцию не передаются.

Вторая функция в качестве аргументов получает два числа, генерирует и выводит на экран произвольное целое число в заданном диапазоне. Эта функция ничего не возвращает.

Третья функция в качестве аргументов получает два числа, генерирует и выводит на экран произвольное дробное число в заданном диапазоне. Эта функция тоже ничего не возвращает.

В основном коде программы запросите у пользователя режим работы (выход / создать целое число / создать дробное число). В зависимости от выбора пользователя выполните действие. Перед каждым «созданием» числа запрашивайте диапазон для генерации. Продолжайте запрашивать режим работы, пока пользователь не выберет «выход».

Проект «Опрос учеников»

Задайте список учеников класса.

Создайте функцию, которая будет выводить на экран случайную ячейку из списка. После вывода эту ячейку нужно удалить.

В основном коде программы создайте «опрос учеников». Узнайте, нужно ли вызвать следующего ученика или закончить опрос. В зависимости от ответа либо закончите опрос, либо выведите имя ученика на экран. Не забудьте обработать вариант, когда ученики закончатся.

Приложение 3

Краткие организационно-методические рекомендации по организации работы на занятии

«Пользовательские функции и методы работы с ними».

В начале занятия можно повторить название уже известных команд и методов и (по возможности) набрать их в терминале. Задача – показать, что функции, методы и операторы отличаются по цвету в момент набора программы. Нас интересуют пока только функции: перечислите их и вспомните, что они делают.

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.

Во время просмотра ролик можно останавливать, чтобы убедиться в том, что информация понятна ученикам. Помимо «быки и коровы» можно открыть и другие проекты (например «телефонный справочник») и проанализировать – как бы можно было поделить программу на разделы.

Особое внимание уделите терминологии: глобальные переменные, локальные, аргументы. Использование в проекте функций – это фактически распределение ролей.

Можете предложить ролевую игру, где есть Управляющий Цикл (один ребенок) и несколько Функций (ребята, выполняющие разные действия). Мораль – если не привлекать помощников, то Управляющий Цикл должен будет все делать сам!