Школа:		
Дата:	**ФИО учителя:**	
Класс:	**Участвовали:**	**Не участволвали:**
Тема урока: Последовательности: строки, списки, словари		
Цели обучения, которые достигаются на данном уроке	Познакомить с конструкцией последовательности: строки, списки, словари Показа принципы работы последовательности: строки, списки, словари	
Цели урока	**Все учащиеся смогут**: • Знать конструкций последовательности: строки, списки, словари и использовать при составлении программ **Большинство учащихся смогут**: • Различать конструкции, работу последовательности: строки, списки, словари и использовать при составлении программ **Некоторые учащиеся смогут:** • Составлять программы с использованием последовательности: строки, списки, словари	
Критерии оценивания	**Владеет** принципами работы последовательности: строки, списки, словари **Умеет** составлять простейшие программы с использованием последовательности: строки, списки, словари	
Воспитание ценностей	Духовное развитие, уважение друг к другу, взаимопонимание	
Предварительные знания	Учащиеся работают над своим уровнем подготовленности к программированию	
Межпредметные связи	математика	
Запланированные этапы урока	**Запланированная деятельность на уроке**	**Ресурсы**
Начало урока **2 мин**	**Организационный момент** **Приветствие учащихся.** **Объявление темы урока, целей обучения, совместное определение целей урока и критериев оценивания**	слайд
Середина урока **10 мин**	**Переход к теме** **Объединение в группы.** **Обсуждение с классом.** - Почему вы объединились именно так? **II. Обобщение и систематизация знаний.** Устный фронтальный опрос с использованием презентации. В языке программирования Python словари (тип	

5 мин
 	
16 мин
	dict) представляют собой еще одну разновидность структур данных наряду со списками и кортежами. *Словарь - это* ***изменяемый*** *(как список)* ***неупорядоченный*** *(в отличие от строк, списков и кортежей) набор элементов "ключ:значение".*
	
	"Неупорядоченный" – значит, что последовательность расположения пар не важна. Язык программирования ее не учитывает, в следствие чего обращение к элементам по индексам невозможно.
	
	В других языках структуры, схожие со словарями, называются по-другому. Например, в Java подобный тип данных называется отображением.
	
	Чтобы представление о словаре стало более понятным, проведем аналогию с обычным словарем, например, англо-русским. На каждое английское слово в таком словаре есть русское слово-перевод: cat – кошка, dog – собака, table – стол и т. д. Если англо-русский словарь описать с помощью Python, то английские слова можно сделать ключами, а русские – их значениями:
	
	`{'cat': 'кошка', 'dog': 'собака', 'bird': 'птица', 'mouse': 'мышь'}`
	
	Обратите внимание на фигурные скобки, именно с их помощью определяется словарь. Синтаксис словаря на Питоне описывается такой схемой:
	
	 
	
	Часто при выводе словаря последовательность пар "ключ:значение" не совпадает с тем, как было введено:
	
	`>>> a = {'cat': 'кошка', 'dog': 'собака', 'bird': 'птица', 'mouse': 'мышь'}`
	`>>> a`
	`{'dog': 'собака', 'cat': 'кошка', 'bird': 'птица', 'mouse': 'мышь'}`
	
	Поскольку в словаре не важен порядок пар, то интерпретатор выводит их так, как ему удобно. Тогда как получить доступ к определенному	

5 мин	элементу, если индексация не возможна в принципе? В словаре доступ к значениям осуществляется по ключам, которые заключаются в квадратные скобки (по аналогии с индексами списков):
	
	`>>> a['cat']`
	`'кошка'`
	`>>> a['bird']`
	`'птица'`
	
	Словари, как и списки, являются изменяемым типом данных: позволительно изменять, добавлять и удалять элементы (пары "ключ:значение"). Изначально словарь можно создать пустым (например, d = {}) и потом заполнить его элементами. Добавление и изменение имеет одинаковый синтаксис: словарь[ключ] = значение. Ключ может быть как уже существующим (тогда происходит изменение значения), так и новым (происходит добавление элемента словаря). Удаление элемента осуществляется с помощью встроенной оператора del языка Python.
	
	`>>> a['elephant'] = 'бегемот' # добавляем`
	`>>> a['table'] = 'стол' # добавляем`
	`>>> a`
	`{'dog': 'собака', 'cat': 'кошка', 'mouse': 'мышь',`
	`'bird': 'птица', 'table': 'стол', 'elephant': 'бегемот'}`
	`>>> a['elephant'] = 'слон' # изменяем`
	`>>> del a['table'] # удаляем`
	`>>> a`
	`{'dog': 'собака', 'cat': 'кошка', 'mouse': 'мышь',`
	`'bird': 'птица', 'elephant': 'слон'}`
	
	В словаре не может быть двух элементов с одинаковыми ключами. Однако могут быть одинаковые значения у разных ключей.
	
	Ключом может быть любой неизменяемый тип данных. Значением – любой тип данных. Значения словарей вполне могут быть структурами, например, другими словарями или списками.
	
	`>>> nums = {1: 'one', 2: 'two', 3: 'three'}`
	`>>> person = {'name': 'Tom', 1: [30, 15, 16], 2: 2.34, ('ab', 100): 'no'}`
	
	Перебор элементов словаря в цикле for	

Элементы словаря перебираются в цикле for также, как элементы других сложных объектов. Однако "по-умолчанию" извлекаются только ключи:

```
>>> nums
{1: 'one', 2: 'two', 3: 'three'}
>>> for i in nums:
...     print(i)
...
1
2
3
```

Но по ключам всегда можно получить значения:

```
>>> for i in nums:
...     print(nums[i])
...
one
two
three
```

С другой стороны у словаря как класса есть метод items(), который создает особую структуру, состоящую из кортежей. Каждый кортеж включает ключ и значение:

```
>>> n = nums.items()
>>> n
dict_items([(1, 'one'), (2, 'two'), (3, 'three')])
```

В цикле for можно распаковывать кортежи, таким образом сразу извлекая как ключ, так и его значение:

```
>>> for key, value in nums.items():
...     print(key, 'is', value)
...
1 is one
2 is two
3 is three
```

Методы словаря keys() и values() позволяют получить отдельно перечни ключей и значений. Так что если, например, надо перебрать только значения или только ключи, лучше воспользоваться одним из этих методов:

```
>>> v_nums = []
>>> for v in nums.values():
...     v_nums.append(v)
```

```
...
>>> v_nums
['one', 'two', 'three']
```

Методы словаря

Кроме рассмотренных выше трех методов items(), keys() и values() словари обладают еще восемью. Это методы clear(), copy(), fromkeys(), get(), pop(), popitem(), setdefault(), update().

Метод clear() удаляет все элементы словаря, но не удаляет сам словарь. В итоге остается пустой словарь:

```
>>> a
{'dog': 'собака', 'cat': 'кошка', 'mouse': 'мышь',
'bird': 'птица', 'elephant': 'слон'}
>>> a.clear()
>>> a
{}
```

Словарь – это изменяемый тип данных. Следовательно, как и список он передается в функцию по ссылке. Поэтому иногда, чтобы избежать нежелательного изменения глобального словаря его копируют. Это делают и с другими целями.

```
>>> nums2 = nums.copy()
>>> nums2[4] = 'four'
>>> nums
{1: 'one', 2: 'two', 3: 'three'}
>>> nums2
{1: 'one', 2: 'two', 3: 'three', 4: 'four'}
```

Метод fromkeys() позволяет создать словарь из списка, элементы которого становятся ключами. Применять метод можно как классу dict, так и к его объектам:

```
>>> a = [1, 2, 3]
>>> c = dict.fromkeys(a)
>>> c
{1: None, 2: None, 3: None}
>>> d = dict.fromkeys(a, 10)
>>> d
{1: 10, 2: 10, 3: 10}
>>> c
{1: None, 2: None, 3: None}
```

Метод get() позволяет получить элемент по его

	ключу:
	
	`>>> nums.get(1)`
	`'one'`
	
	Равносильно nums[1].
	
	Метод pop() удаляет из словаря элемент по указанному ключу и возвращает значение удаленной пары. Метод popitem() не принимает аргументов, удаляет и возвращает произвольный элемент.
	
	`>>> nums.pop(1)`
	`'one'`
	`>>> nums`
	`{2: 'two', 3: 'three'}`
	`>>> nums.popitem()`
	`(2, 'two')`
	`>>> nums`
	`{3: 'three'}`
	
	С помощью setdefault() можно добавить элемент в словарь:
	
	`>>> nums.setdefault(4, 'four')`
	`'four'`
	`>>> nums`
	`{3: 'three', 4: 'four'}`
	
	Равносильно nums[4] = 'four', если элемент с ключом 4 отсутствует в словаре. Если он уже есть, то nums[4] = 'four' перезапишет старое значение, setdefault() – нет.
	
	С помощью update() можно добавить в словарь другой словарь:
	
	`>>> nums.update({6: 'six', 7: 'seven'})`
	`>>> nums`
	`{3: 'three', 4: 'four', 6: 'six', 7: 'seven'}`
	
	Также метод обновляет значения существующих ключей. Включает еще ряд особенностей.
	
	Практическая работа
	
	1. Создайте словарь, связав его с переменной school, и наполните данными, которые бы отражали количество учащихся в разных классах	

	(1а, 1б, 2б, 6а, 7в и т. п.). Внесите изменения в словарь согласно следующему: а) в одном из классов изменилось количество учащихся, б) в школе появился новый класс, с) в школе был расформирован (удален) другой класс. Вычислите общее количество учащихся в школе. 2. Создайте словарь, где ключами являются числа, а значениями – строки. Примените к нему метод items(), полученный объект dict_items передайте в написанную вами функцию, которая создает и возвращает новый словарь, "обратный" исходному, т. е. ключами являются строки, а значениями – числа.	
Конец урока **<u>39-40</u> мин**	**Рефлексия.** Ученики анализируют деятельность на уроке, описывают затруднения, предлагают пути их преодоления.	
Дифференциация – каким образом Вы планируете оказать больше поддержки? Какие задачи Вы планируете поставить перед более способными учащимися?	**Оценивание – как Вы планируете проверить уровень усвоения материала учащимися?**	**Охрана здоровья и соблюдение техники безопасности**
Дифференциация в подборе заданий, в ожидаемом результате от конкретного ученика, в оказании индивидуальной поддержки учащемуся на этапе решении задач.	Взаимооценивание (по результатам эксперимента) Самооценивание (решение задач)	Соблюдение Правил техники безопасности в кабинете информатики