Словарь — неупорядоченная структура данных, которая позволяет хранить пары «ключ — значение». Структура данных, позволяющая идентифицировать ее элементы не по числовому индексу, а по произвольному, называется *словарем* или *ассоциативным массивом*. Соответствующая структура данных в языке Питон называется dict.

Вот примеры создания и использования словаря на Python:

Пример 1:	
Capitals={}	#Создадим словарь с помощью литерала
Capitals = dict()	# Создадим пустой словать Capitals
Capitals['Russia'] = 'Moscow' Capitals['Ukraine'] = 'Kiev' Capitals['USA'] = 'Washington'	# Заполним его несколькими значениями
Countries = ['Russia', 'France', 'USA', 'Russia']	#Создадим список стран

for country in Countries:
  if country in Capitals:
  print('Столица страны ' + country + ': ' + Capitals[country])
  else:
  print('В базе нет страны с названием ' + country)
	# Для каждой страны из списка проверим, есть ли она в словаре Capitals

Пример 2:	
dictionary = {'персона': 'человек', 'марафон': 'гонка бегунов длиной около 26 миль', 'противостоять': 'оставаться сильным, несмотря на давление', 'бежать': 'двигаться со скоростью'}	#Создаем и сразу заполняем словарь значениями, он использует строки в качестве ключей и значений
Пример 3:	
gender_dict = {0: 'муж', 1: 'жен'}	ключами являются числа, а значениями — строки

Методы и основные функции	
dictionary['марафон']	# берём значение с ключом "марафон" Для получения значения конкретного ключа используются квадратные скобки []. Предположим, что в нашем словаре есть пара 'марафон': 26.
dictionary['туфля'] = 'род обуви, закрывающей ногу не выше щиколотки'	# **Добавляем** ключ "туфля" со значением "род обуви, закрывающей ногу не выше щиколотки"
del dictionary['противостоять']	# Удаляем значение с ключом "противостоять" из словаря

dictionary**.update**({'бежал': 'бежать в прошедшем времени', 'туфли': 'туфля во множественном числе'})	# Добавляем две пары в словарь dictionary, используя метод update
story_count = {'сто': 100, 'девяносто': 90, 'двенадцать': 12, 'пять': 5} story_count.**get**('двенадцать')	# Допустим, у нас есть словарь story_count, Ключ «двенадцать» существует и метод get в данном случае вернёт 12 ! Если указанного ключа не существует, метод вернёт None.

>>> story_count**.pop**('девяносто')
90
>>> story_count
{'двенадцать': 12, 'сто': 100, 'пять': 5}
	# pop() удаляет ключ и возвращает соответствующее ему значение.

>>> story_count.**keys**() ['сто', 'пять', 'двенадцать']	# keys() возвращает коллекцию ключей в словаре.
>>> story_count.**values**() [100, 12, 5]	# values() возвращает коллекцию значений в словаре.
>>> dictionary.**items**() [('персона', 'человек'), ('бежать', 'двигаться со скоростью'), ('туфля', 'род обуви, закрывающей ногу не выше щиколотки'), ('бежал', 'бежать в прошедшем времени'), ('марафон', 'гонка бегунов длиной около 26 миль'), ('туфли', 'туфля во множественном числе')]	# items() возвращает пары «ключ — значение»
for key in story_count: **print(key)**	# можно провести итерацию по каждому ключу в словаре, вместо story_count можно использовать story_count.keys()
>>> **for key, value in dictionary.items():** **print(key, value)** ('персона', 'человек') ('бежать', 'двигаться со скоростью') ('туфля', 'род обуви, закрывающей ногу не выше щиколотки') ('бежал', 'бежать в прошедшем времени') ('марафон', 'гонка бегунов длиной около 26 миль') ('туфли', 'туфля во множественном числе')	# цикл for использует метод items() для получения пары «ключ — значение» на каждую итерацию