Строки (тип str) — набор символов, заключенных в кавычки (например, "ball", 'dkfjUUv', '6589'). Примечание: кавычки в Python могут быть одинарными или двойными; одиночный символ в кавычках также является строкой, отдельного символьного типа в Питоне нет.

Строки – это простой тип данных, т.е. это не структура данных как, например, List. Строки являются упорядоченной последовательностью элементов и могут быть проиндексированы. Индексация строк начинается с нуля. Из строки можно извлекать отдельные символы и срезы (работа со срезами описана в теме «Списки»). Индексы строк также могут быть указаны отрицательными числами. В этом случае индексирование начинается с конца строки: -1 относится к последнему символу, -2 к предпоследнему и так далее. Попытка обращения по индексу меньшему чем -len(s), приводит к ошибке Index Error.

```
Раздаточный материал № 71
>>> s = "Hello, World!"
>>> s[0]
'H'
>>> s[7:]
'World!'
>>> s[::2] # здесь извлечение идет с шагом = 2
'Hlo ol!'
```

Важным отличием от списков является неизменяемость строк. Нельзя перезаписать какой-то отдельный символ или срез в строке (ошибке TypeError). Если требуется изменить строку, то следует создать новую из срезов старой.

Раздаточный материал № 72

```
>>> s = s[0:-1] + '.'
>>> s
'Hello, World.' # старое значение s теряется
```

Python 3 поддерживает ASCII, Unicode, в том числе позволяет использовать символы Unicode в строках.

Методы и функции строк можно посмотреть по команде dir(str), получить информацию по каждому – help(str.имя метода).

Методы похожи на функции. Метод — специализированный тип вызываемой процедуры, тесно связанный с объектом. Как и функция, метод вызывается для выполнения отдельной задачи, но он вызывается только вместе с определенным объектом и знает о нем во время выполнения.

Синтаксис для вызова метода объекта выглядит следующим образом:

```
Раздаточный материал № 73 obj.foo(<args>)
```

Этот код вызывает метод .foo() объекта obj. <args> — аргументы, передаваемые методу (если есть).

После работы метода возвращается копия исходного объекта с выполненными изменениями. Исходный объект не изменяется.

Раздаточный материал № 74 (справочно)

Строковые операторы		
+ - конкатенация строк		>>> s = 'py' >>> t = 'th'

>>> u = 'on'   >>> s + t + u   python'   >>> print('Привст, ' + 'Мир!')   >>> s + t + u   python'   >>> print('Привст, ' + 'Мир!')   >>> s + y + u   python'   >>> s = 'Pyt.   >>> s + s + v   >>> s = 'Pyt.   >>> s + s + v   >>> s + y   >> s   y   >> s + y   >> s   y   >> s + y   >> s   y   >>		
* - умпожение строк. Значение множительным числом		>>> u = 'on'
** - умпожение строк. Значение множителя должно быть целым положительным числом   ** - умпожента строк. Значение множительным числом   ** - умпожента гине, если подстрока входит в строку, и False, если нет.   ** Ertъ также оператор пот in, у которого обратная логика   ** Erbocumuse функции строк   ** >> s in Tlove Java.'   ** False   **		>>> s + t + u
# - умножение строк. Значение множителя должно быть целым положительным учислом    10		'python'
должно быть целым положительным числом  по - оператор принадлежности подстроки, возвращает Ттие, если подстрока входит в строку, и False, если нет.  Есть также оператор пот in, у которого обратная логика  Ветроенные функции строк  Ветроенные функции строк  Ветроенные функции строк  зумка принимать необязательный дарумент-строку, указывающей по какому символу или подстроке следует выполнить разделение  Может принимать необязательный дарумент-строку, указывающей по какому символу или подстроке следует выполнить разделение  Метод строк јоіп() выполняет обратное действие. Оп формирует из списка строку, Поскольку это метод строки, то впереди ставится строка-разделитель, а в скобках — передается список. Если разделитель не нужен, то метод применяется к пустой строке во восей строке, а плипь на каком-то ее отрезке. В этом случае указывается первый и последний индексы отрезка. Если роследний индексы отрезка. Если последний индексы отрезка. Если роследний индексы отрезка. Если роследний индексы отрезка. Если горки. Метод find() возвращает только первое вхождение.   лерействие Оп формирует из списка строку, последний индексы отрезка. В сли последний индексы отрезка. Если роследний индексы отрезка. Если горки. Метод find() возвращает только первое вхождение.  лерействие Оп формирует из списка строку, последний индексы отрезка. Если горки подстроку в строке и возвращает только первое вхождение.  лерействие Оп формирует из списка строку, последний индексы отрезка. Если горки подстроку в строке и подстроку в геф blue orange white'  >>> s. find('blue')  >>> s. find('green')  ->> s. find('green')  ->> s. find('green')  ->> s. find('DA', 0, 6)  ->> letters.find('DA', 0, 6)  ->> letters.replace('DA', 'NET')		>>> print('Привет, ' + 'Мир!')
то поератор принадлежности подстрока возращает Тие, если подстрока входит в строку, и False, если нет.    Встроку, и False, если нет.	* - умножение строк. Значение множителя	>>> s = 'py.'
in - оператор принадлежности подстроки, возвращает Тпе, если подстрока входит в строку, и False, если нет.    Eсть также оператор not in, у которого обратная логика   Встроенные функции строк   False   >>> s in 'I love Python.' True   >>> s in 'I love Java.'   False   >>> s in 'I love Java.'   Salva.'   S	должно быть целым положительным	>>> s * 4
in - оператор принадлежности подстроки, возвращает Тпе, если подстрока входит в строку, и False, если нет.    Eсть также оператор not in, у которого обратная логика   Встроенные функции строк   False   >>> s in 'I love Python.' True   >>> s in 'I love Java.'   False   >>> s in 'I love Java.'   Salva.'   S	числом	'py.py.py.py.'
возвращает Ťrue, если подстрока входит в строку, и False, если нет.  Есть также оператор not in, у которого обратная логика  Ветроенные функции строк  дов.  Ветроенные функции строк  ворина функции строк  разделение  Ветроенные функции строк  гed blue orange white  >>> s = input()  red blue orange white'  >>> s = s, split()  >>> s    red blue orange white'  >>> s    red blue orange whi	in - оператор принадлежности подстроки,	
тине подстроки. В сторок и подеторок и подеторок и подеторок. В том сторок и подеторок. В том сторок и подеторок. В том сторок и подеторок и подеторок. В том сторок и подеторок и подеторок. В том сторок и подеторок и под		T
Set	-	<u> </u>
False	± • • • • • • • • • • • • • • • • • • •	>>> s in 'I love Java.'
ярін() позволяєт разбить строку по пробелам. В результате получаєтся список слов.  Может принимать необязательный аргумент-строку, указывающей по какому символу или подстроке следует выполнить разделение  метод строк join() выполняет обратное действие. Оп формирует из списка строку. Поскольку это метод строки, то впереди ставится строка-разделитель, а в скобках — передается список.  Если разделитель не нужен, то метод применяется к пустой строке в біли() ищет подстроку в строке и возвращает тидекс первого элемента найдена, то возвращает -1. Поиск может производиться не во всей строке, а лишь на каком-то ео стрезке. В этом случає указываєтся первый и последний индексы отрезка. Если подледний индексы отрезка. Если последний индексы отрезка строку. № 1 № 1 № 1 № 1 № 1 № 1 № 1 № 1 № 1 №		
split()         позволяет разбить строку по поробелам. В результате получается список слов.         >>> s = input() red blue orange white           может принимать необязательный аргумент-строку, указывающей по какому символу или подстроке следует выполнить разделение         >>> s           метод строк join() выполняет обратное действие. Он формирует из списка строку. Поскольку это метод строки, то впереди ставится строка-разделитель, а в скобках — передается список. Если разделитель не нужен, то метод применяется к пустой строке         >>> s'red blue orange white'           find() ищет подстроку в строке и возвращает идскс первого элемента найденной подстроки. Если подстрока не найдена, то возвращает -1. Поиск может производиться не во всей строке, а лишь на каком-то еотрезке. В этом случае указывается нервый и последний индексы отрезка. Если последний индексы отрезка. Если последний не указан, то ищется до конща строки. Метод find() возвращает только первое вхождение.         >>> s. sinut() red blue orange white'           >>>> ".'.join(sl)         >>> ".join(sl)           'red blue orange white'         >>> ".join(sl)           'red-blue-orange-white'         >>> ".join(sl)           'red-blueorangewhite'         >>> s. find('blue')           >>> ".join(sl)         >>> s. s. find('blue')           >>> s. s. find('green')         >>> s. s. find('green')           >>> letters = 'ABCDACFDA'         >>> letters = 'ABCDACFDA'           >>> letters. find('A', 3)         >>> letters. find('DA', 0, 6)           3 # Поиск идет с третьего идекса и до конца, а также с перв	1	
пробелам. В результате получается список слов.  Может принимать необязательный артумент-строку, указывающей по какому символу или подстроке следует выполнить разделение  ——————————————————————————————————		ř
слов.  Может принимать необязательный аргумент-строку, указывающей по какому символу или подстроке следует выполнить разделение  метод строк join() выполняет обратное действие. Он формирует из списка строку. Поскольку это метод строки, то впереди ставится строка-разделитель, а в скобках — передается список. Если разделитель не нужен, то метод применяется к пустой строке  find() ищет подстроку в строке и возвращает индек первого элемента найдению подстроки. Если подстрока не найдена, то возвращает -1. Поиск может производиться не во всей строке, а лишь на каком-то ес отрезке. В этом случае указывается первый и последний индексы отрезка. Если подстрий индексы отрезка. Если последний не указан, то ищется до конца, то конца, а также с первого и до шестого  replace() заменяет одну подстроку на  >>> s red blue orange white'  >>> s.split('e') [r', 'd blu', 'orang', 'whit', "]  >>> s 'red blue orange.    'red, 'blue', 'orange', 'white']  >>> s 'red blue orange.    'red, 'blue', 'orange', 'white']  >>> s 'red blue orange.    'red, 'blue', 'orange', 'white']  >>> s 'red blue orange.    'red, 'blue', 'orange', 'white']  >>> s 'red blue orange.    'red-blue-orange-white'    'red-blue-orange-white'    'red-blue-orange-white'    'red-blue-orange-white'    'red-blue-orange-white'    'red-blue-orange white'  >>> s.find('blue')  4  >>> s.find('blue')  4  >>> s.find('green')  -1  -1  -1  -1  -1  -1  -1  -1  -1  -	* · · ·	± "
может принимать необязательный аргумент-строку, указывающей по какому символу или подстроке следует выполнить разделение  ——————————————————————————————————		
аргумент-строку, указывающей по какому символу или подстроке следует выполнить разделение  >>> sl = s.split() >>> sl ['red', 'blue', 'orange', 'white'] >>> s 'red blue orange white'  >>> s.split('e') ['r', 'd blu', 'orang', 'whit', "] >>> '40030023'.split('00') ['4', '3', '23']  Метод строк join() выполняет обратное действие. Он формирует из списка строку. Поскольку это метод строки, то впереди ставится строка-разделитель, а в скобках — передается список. Если разделитель не нужен, то метод применяется к пустой строке find() ищет подстроку в строке и возвращает индекс первого элемента найденаю подстроки. Если подстрока не найдена, то возвращает -1. Поиск может производиться не во всей строке, а лишь на каком-то ее отрезке. В этом случае указывается первый и последний индексы отрезка. Если последний не указан, то ищется до конца строки. Метод find() возвращает только первое вхождение.  >>> sl = s.split() >>> sl ['red', 'blue', 'orange', 'white'] >>> s.y split('e') ['r', 'd blu', 'orange', 'white'] >>> s.split('e') ['r', 'd blu', 'orange', 'white'] >>> s.split('e') ['r', 'd blu', 'orange', 'white'] >>> s.y split('e') ['r', 'd blu', 'orange', 'white'] >>> s.y split('e') ['r', 'd blu', 'orange', 'white'] >>> s.split('e') ['r', 'd blu', 'orange', 'white'] >>> s.split('e') ['r', 'd blu', 'orange', 'white'] >>> s.split('e') ['r', 'd blu', 'orange', 'white'] >>> s.y split('e') ['r', 'd blu', 'orange', 'white'] >>> s.split(e') ['r', 'd blu', 'orange', 'white'] >>> s.y ide blue orange white' >>> s.split('e') ['r', 'd blu', 'orange', 'white'] >>> '-'join(sl) 'red-blue-orange-white' >>> s.split('e') ['r', 'd blu', 'orange', 'white'] >>> '-'join(sl) 'red-blue-orange-white' >>> s.split('e')   'red-		
Section   Sec	1	
разделение  ['red', 'blue', 'orange', 'white']  >>> s		- v
>>> s 'red blue orange white' >>> s.split('e') ['r', 'd blu', 'orang', 'whit', "] >>> '40030023'.split('00') ['4', '3', '23']  Метод строк join() выполняет обратное действие. Он формирует из списка строку. Поскольку это метод строки, то впереди ставится строка-разделитель, а в скобках — передается список.  Если разделитель не нужен, то метод применяется к пустой строке  find() ищет подстроку в строке и возвращает индекс первого элемента найденной подстроки. Если подстрока не найдена, то возвращает -1. Поиск может производиться не во всей строке, а лишь на каком-то ее отрезке. В этом случае указывается первый и последний индексы отрезка. Если последний не указан, то ищется до конца строки. Метод find() возвращает только первое вхождение.  >>> s ''.'.join(sl) 'red-blue-orange-white'  >>> ".'.join(sl) 'redblueorangewhite'  >>> s ''id blue orange white'  >>> s.find('blue')  >>> s.find('blue')  >>> s.find('green')  >>> letters = 'ABCDACFDA'  >>> letters.find('A', 3)  4  >>> letters.find('DA', 0, 6)  3 #Поиск идет с третьего индекса и до конца, а также с первого и до шестого  **Peplace() заменяет одну подстроку на  >>> letters.replace('DA', 'NET')	l • • • • • • • • • • • • • • • • • • •	
'red blue orange white'  >>> s.split('e') [T', 'd blu', 'orang', 'whit', "]  >>> '40030023'.split('00') ['4', '3', '23']  Метод строк join() выполняет обратное действие. Он формирует из списка строку. Поскольку это метод строки, то впереди ставится строка-разделитель, а в скобках — передается список.  Если разделитель не нужен, то метод применяется к пустой строке  find() ищет подстроку в строке и возвращает индекс первого элемента найденной подстроки. Если подстрока не найдена, то возвращает -1. Поиск может производиться не во всей строке, а лишь на каком-то ее отрезке. В этом случае указывается первый и последний индексы отрезка. Если последний не указан, то ищется до конща строки. Метод find() возвращает только первое вхождение.  'red blue-orange-white'  >>> s''.join(sl)  'red-blue-orange-white'  >>> s''.join(sl)  'red blue orange white'  >>> s.find('blue')  >>> s.find('blue')  >>> s.find('green')  -1  -1  -2  >>> letters = 'ABCDACFDA'  >>> letters.find('DA', 0, 6)  3  #Поиск идет с третьего индекса и до конца, а также с первого и до шестого  >>> letters.replace('DA', 'NET')	разделение	
>>> s.split('e')   ['r', 'd blu',' orang', 'whit', "]   >>> 40030023'.split('00')   ['4', '3', '23']     >>> '-'.join(sl)     'red-blue-orange-white'		
['r', 'd blu', ' orang', 'whit', '']   >>> '40030023'.split('00')   ['4', '3', '23']		red blue blange willte
['r', 'd blu', ' orang', 'whit', '']   >>> '40030023'.split('00')   ['4', '3', '23']		>>> c cplit('a')
>>> '40030023'.split('00') ['4', '3', '23']  Метод строк join() выполняет обратное действие. Он формирует из списка строку. Поскольку это метод строки, то впереди ставится строка-разделитель, а в скобках — передается список.  Если разделитель не нужен, то метод применяется к пустой строке  find() ищет подстроку в строке и возвращает индекс первого элемента найденной подстроки. Если подстрока не найдена, то возвращает -1. Поиск может производиться не во всей строке, а лишь на каком-то ее отрезке. В этом случае указывается первый и последний индексы отрезка. Если последний и нуказан, то ищется до конца строки. Метод find() возвращает только первое вхождение.  >>> '40030023'.split('00') ['4', '3', '23']  >>> '-'.join(sl)  'red-blue-orange-white'  >>> s''.join(sl)  'redblueorangewhite'  >>> s s'ind('blue')  4  >>> s.find('blue')  4  >>> s.find('green')  -1  -1  >>> letters = 'ABCDACFDA'  >>> letters.find('DA', 3)  4  >>> letters.find('DA', 0, 6)  3  # Поиск идет с третьего индекса и до конца, а также с первого и до шестого  **replace() заменяет одну подстроку на  >>> letters.replace('DA', 'NET')		
Метод строк join() выполняет обратное действие. Он формирует из списка строку. Поскольку это метод строки, то впереди ставится строка-разделитель, а в скобках — передается список.  Если разделитель не нужен, то метод применяется к пустой строке  find() ищет подстроку в строке и возвращает индекс первого элемента найденной подстроки. Если подстрока не найдена, то возвращает -1. Поиск может производиться не во всей строке, а лишь на каком-то ее отрезке. В этом случае указывается первый и последний индексы отрезка. Если подследний индексы отрезка. Если последний индексы отрезка. Если последний индексы отрезка. Если последний не указан, то ищется до конца строки. Метод find() возвращает только первое вхождение.  14', '3', '23']  >>> '-'.join(sl)  'red-blue-orange-white'  >>> s' '-i.join(sl)  'red-blue-orange-white'  >>> s' '-i.join(sl)  'red-blue-orange-white'  >>> s' '-i.join(sl)  'red-blue-orange-white'  >>> s' '-i.join(sl)  'red-blue-orange-white'		
Метод строк join() выполняет обратное действие. Он формирует из списка строку. Поскольку это метод строки, то впереди ставится строка-разделитель, а в скобках — передается список.  Если разделитель не нужен, то метод применяется к пустой строке  find() ищет подстроку в строке и возвращает индекс первого элемента найденной подстроки. Если подстрока не найдена, то возвращает -1. Поиск может производиться не во всей строке, а лишь на каком-то ее отрезке. В этом случае указывается первый и последний индексы отрезка. Если последний индексы отрезка. Если последний индексы отрезка. Если последний индексы отрезка. Если последний не указан, то ищется до конца строки. Метод find() возвращает только первое вхождение.  →>> '-'.join(sl)  'red-blue-orange-white'  >>> ".join(sl)  'redblueorangewhite'  >>> s.find('blue')  4  >>> s.find('blue')  4  >>> letters = 'ABCDACFDA'  >>> letters.find('A', 3)  4  >>> letters.find('DA', 0, 6)  3  # Поиск идет с третьего индекса и до конца, а также с первого и до шестого  replace() заменяет одну подстроку на  >>> letters.replace('DA', 'NET')		± ', '/
терименяется строка не нужен, то метод применяется к пустой строке  подстроки. Если подстроку в строке и возвращает индекс первого элемента найденной подстроки. Если подстрока не найдена, то возвращает -1. Поиск может производиться не во всей строке, а лишь на каком-то ее отрезке. В этом случае указывается первый и последний индексы отрезка. Если подстроки. Метод find() возвращает только первое вхождение.    Ted-blue-orange-white	Morrow orner icin() prygot ofnerwae	
Поскольку это метод строки, то впереди ставится строка-разделитель, а в скобках — передается список.  Если разделитель не нужен, то метод применяется к пустой строке  find() ищет подстроку в строке и возвращает индекс первого элемента найденной подстроки. Если подстрока не найдена, то возвращает -1. Поиск может производиться не во всей строке, а лишь на каком-то ее отрезке. В этом случае указывается первый и последний индексы отрезка. Если последний не указан, то ищется до конца строки. Метод find() возвращает только первое вхождение.  1		, ,
тередается список.  Если разделитель не нужен, то метод применяется к пустой строке  find() ищет подстроку в строке и возвращает индекс первого элемента найдена, то возвращает -1. Поиск может производиться не во всей строке, а лишь на каком-то ее отрезке. В этом случае указывается первый и последний индексы отрезка. Если последний индексы отрезка. Если последний не указан, то ищется до конца строки. Метод find() возвращает только первое вхождение.  1		red-blue-orange-winte
передается список.  Если разделитель не нужен, то метод применяется к пустой строке  find() ищет подстроку в строке и возвращает индекс первого элемента найденной подстроки. Если подстрока не найдена, то возвращает -1. Поиск может производиться не во всей строке, а лишь на каком-то ее отрезке. В этом случае указывается первый и последний индексы отрезка. Если последний не указан, то ищется до конца строки. Метод find() возвращает только первое вхождение.		
Eсли разделитель не нужен, то метод применяется к пустой строке   find() ищет подстроку в строке и возвращает индекс первого элемента найденной подстроки. Если подстрока не найдена, то возвращает -1. Поиск может производиться не во всей строке, а лишь на каком-то ее отрезке. В этом случае указывается первый и последний индексы отрезка. Если последний не указан, то ищется до конца строки. Метод find() возвращает только первое вхождение.   4	* * * * * * * * * * * * * * * * * * *	>>> "!:a::r(a1)
применяется к пустой строке	-	
find() ищет подстроку в строке и возвращает индекс первого элемента найденной подстроки. Если подстрока не найдена, то возвращает -1. Поиск может производиться не во всей строке, а лишь на каком-то ее отрезке. В этом случае указывается первый и последний индексы отрезка. Если последний не указан, то ищется до конца строки. Метод find() возвращает только первое вхождение.         >>> s.find('blue')           *** *** *** *** *** *** *** *** *** **	· · · · · · · · · · · · · · · · · · ·	reddiueorangewnite
индекс первого элемента найденной подстроки. Если подстрока не найдена, то возвращает -1. Поиск может производиться не во всей строке, а лишь на каком-то ее отрезке. В этом случае указывается первый и последний индексы отрезка. Если последний не указан, то ищется до конца строки. Метод find() возвращает только первое вхождение.       **>> letters = 'ABCDACFDA'         **>>> letters.find('A', 3)         4       **>>> letters.find('DA', 0, 6)         3       # Поиск идет с третьего индекса и до конца, а также с первого и до шестого         ***replace() заменяет одну подстроку на       **>> letters.replace('DA', 'NET')		
подстроки. Если подстрока не найдена, то возвращает -1. Поиск может производиться не во всей строке, а лишь на каком-то ее отрезке. В этом случае указывается первый и последний индексы отрезка. Если последний не указан, то ищется до конца строки. Метод find() возвращает только первое вхождение.       >>> s.find('blue')         4       >>> s.find('green')         -1       >>> letters = 'ABCDACFDA'         >>> letters.find('A', 3)       4         >>> letters.find('DA', 0, 6)       3         # Поиск идет с третьего индекса и до конца, а также с первого и до шестого         replace() заменяет одну подстроку на       >>> letters.replace('DA', 'NET')	1	
возвращает -1. Поиск может производиться не во всей строке, а лишь на каком-то ее отрезке. В этом случае указывается первый и последний индексы отрезка. Если последний не указан, то ищется до конца строки. Метод find() возвращает только первое вхождение.  ———————————————————————————————————	<u> </u>	
не во всей строке, а лишь на каком-то ее отрезке. В этом случае указывается первый и последний индексы отрезка. Если последний не указан, то ищется до конца строки. Метод find() возвращает только первое вхождение.  ———————————————————————————————————		`
отрезке. В этом случае указывается первый и последний индексы отрезка. Если последний не указан, то ищется до конца строки. Метод find() возвращает только первое вхождение.  ———————————————————————————————————		
и последний индексы отрезка. Если последний не указан, то ищется до конца строки. Метод find() возвращает только первое вхождение.  ———————————————————————————————————	1	1 .
последний не указан, то ищется до конца строки. Метод find() возвращает только первое вхождение.		-1
строки. Метод find() возвращает только первое вхождение.  >>> letters.find('A', 3)  4  >>> letters.find('DA', 0, 6)  3  # Поиск идет с третьего индекса и до конца, а также с первого и до шестого  replace() заменяет одну подстроку на >>> letters.replace('DA', 'NET')	-	
первое вхождение.  4  >>> letters.find('DA', 0, 6)  3  # Поиск идет с третьего индекса и до конца, а также с первого и до шестого  replace() заменяет одну подстроку на >>> letters.replace('DA', 'NET')	· · · · · · · · · · · · · · · · · · ·	
>>> letters.find('DA', 0, 6) 3 # Поиск идет с третьего индекса и до конца, а также с первого и до шестого  replace() заменяет одну подстроку на >>> letters.replace('DA', 'NET')		1
3 # Поиск идет с третьего индекса и до конца, а также с первого и до шестого replace() заменяет одну подстроку на >>> letters.replace('DA', 'NET')	первое вхождение.	
# Поиск идет с третьего индекса и до конца, а также с первого и до шестого  replace() заменяет одну подстроку на >>> letters.replace('DA', 'NET')		
а также с первого и до шестого  replace() заменяет одну подстроку на >>> letters.replace('DA', 'NET')		1 -
replace() заменяет одну подстроку на >>> letters.replace('DA', 'NET')		_
		•
другую 'ABCNETCFNET'	replace() заменяет одну подстроку на	<b>2</b> , , , , ,
	другую	'ABCNETCFNET'
Исходная строка не меняется:		Исходная строка не меняется:

	>>> letters
	'ABCDACFDA'
	если результат надо сохранить, то его надо
	присвоить переменной
	>>> new_letters = letters.replace('DA', 'NET')
	>>> new_letters
	'ABCNETCFNET'
ord(c) возвращает числовое значение для	>>> ord('a')
заданного символа	97
	>>> ord('#')
	35
chr(n) возвращает символьное значение для	>>> chr(8364)
данного целого числа.	'€'
данного целого числа.	>>> chr(8721)
	'Σ'
1(-)	
len(s) возвращает длину строки	>>> s = 'Простая строка.'
	>>> len(s)
(18)	15
str(obj) возвращает строковое	>>> str(49.2)
представление объекта	'49.2'
	>>> str(3+4j)
	'(3+4j)'
	>>> str(3 + 29)
	'32'
	>>> str('py')
	'py'
Встроенные в	методы строк
string.capitalize() приводит первую букву в	>>> s = 'everyTHing yoU Can IMaGine is
верхний регистр, остальные в нижний.	rEAl'
	>>> s.capitalize()
	'Everything you can imagine is real'
string.lower() преобразует все буквенные	>>> 'everyTHing yoU Can IMaGine is
символы в строчные.	rEAl'.lower()
ommonia a orpo mare.	'everything you can imagine is real'
string.swapcase() меняет регистр буквенных	>>> 'the sun also rises'.title()
символов на противоположный.	'The Sun Also Rises'
символов на противоположный.	The Bull Also Rises
	SSS 'follow we @DVTHOM' title()
	>>> 'follow us @PYTHON'.title()
	'Follow Us @Python'
string.upper() преобразует все буквенные	>>> 'follow us @PYTHON'.upper()
символы в заглавные.	'FOLLOW US @PYTHON'
string.count( <sub>[, <start>[, <end>]])</end></start></sub>	
подсчитывает количество вхождений	
подстроки в строку.	
s.count( <sub>) возвращает количество</sub>	>>> 'foo goo moo'.count('oo')
точных вхождений подстроки <sub> в s:</sub>	3
Количество вхождений изменится, если	>>> 'foo goo moo'.count('oo', 0, 8)
указать <start> и <end></end></start>	2
string.endswith( <suffix>[, <start>[, <end>]])</end></start></suffix>	
определяет, заканчивается ли строка	
заданной подстрокой	>>> 'python'.endswith('on')
63	F J ( 322 )

s.endswith( <suffix>) возвращает, True если s</suffix>	True
заканчивается указанным <suffix> и False</suffix>	>>> 'python'.endswith('or')
если нет	False
Сравнение ограничено подстрокой, между	>>> 'python'.endswith('yt', 0, 4)
<start> и <end>, если они указаны</end></start>	True
·	>>> 'python'.endswith('yt', 2, 4)
	False
string.find( <sub>[, <start>[, <end>]]) ищет в</end></start></sub>	
строке заданную подстроку	
s.find( <sub>) возвращает первый индекс в s</sub>	>>> 'Follow Us @Python'.find('Us')
который соответствует началу строки	7
<sub></sub>	/ >>> 'Follow Us @Python'.find('you')
Этот метод возвращает, -1 если указанная	-1
подстрока не найдена	
±	>>> 'Follow Us @Python'.find('Us', 4)
Поиск в строке ограничивается подстрокой,	7
между <start> и <end>, если они указаны</end></start>	>>> 'Follow Us @Python'.find('Us', 4, 7)
C 1( , 1, )	-1
s.rfind( <sub>) возвращает индекс</sub>	>>> 'Follow Us @Python'.rfind('o')
последнего вхождения подстроки <sub> в s,</sub>	15
который соответствует началу <sub>. Как и</sub>	
в .find(), если подстрока не найдена,	
возвращается -1. Поиск в строке	
ограничивается подстрокой, между <start></start>	
и <end>, если они указаны.</end>	
string.isalnum() определяет, состоит ли	>>> 'abc123'.isalnum()
строка из букв и цифр, возвращает True,	True
если строка в не пустая, а все ее символы	>>> 'abc\$123'.isalnum()
буквенно-цифровые (либо буква, либо	False
цифра). В другом случае False	>>> ".isalnum()
Hippay. B Apyrom only fact talse	False
string.isalpha() определяет, состоит ли	>>> 'ABCabc'.isalpha()
строка только из букв, возвращает True,	True
если строка в не пустая, а все ее символы	>>> 'abc123'.isalpha()
буквенные. В другом случае False	False
string.isdigit() определяет, состоит ли строка	
	>>> '123'.isdigit()
из цифр (проверка на число), возвращает	True
Ттие когда строка в не пустая и все ее	>>> '123abc'.isdigit()
символы являются цифрами, а в False если	False
HET	16 201:11 16 0
string.isidentifier() определяет, является ли	>>> 'foo32'.isidentifier()
строка допустимым идентификатором	True
Python, возвращает True, если s валидный	>>> '32foo'.isidentifier()
идентификатор (название переменной,	False
функции, класса и т.д.) python, а в False если	>>> 'foo\$32'.isidentifier()
нет. Вернет True для строки, которая	False
соответствует зарезервированному	
ключевому слову python, даже если его	
нельзя использовать	
string.islower() определяет, являются ли	>>> 'abc'.islower()
буквенные символы строки строчными,	True
возвращает True, если строка s не пустая, и	>>> 'abc1\$d'.islower()
все содержащиеся в нем буквенные	True
Sometiment is included the contraction of th	1

символы строчные, а False если нет. Не	>>> 'Abc1\$D'.islower()
алфавитные символы игнорируются	False
string.isprintable() определяет, состоит ли	>>> 'a\tb'.isprintable() # \t - символ
строка только из печатаемых символов,	табуляции
возвращает, True если строка s пустая или	False
все буквенные символы которые она	>>> 'a b'.isprintable()
содержит можно вывести на экран.	True
Возвращает, False если s содержит хотя бы	>>> ".isprintable()
один специальный символ. Не алфавитные	True
символы игнорируются. Это единственный	>>> 'a\nb'.isprintable() # \n - символ
метод, который возвращает True, если s	перевода строки
пустая строка. Все остальные	False
возвращаются False	
string.isspace() определяет, состоит ли	>>> '\t \n '.isspace()
строка только из пробельных символов,	True
возвращает True, если s не пустая строка, и	>>> ' a '.isspace()
все символы являются пробельными, а	False
False, если нет. Наиболее часто	
встречающиеся пробельные символы — это	
пробел ' ', табуляция '\t' и новая строка '\n'	
string.istitle() определяет, начинаются ли	>>> 'This Is A Title'.istitle()
слова строки с заглавной буквы, возвращает	True
True когда s не пустая строка и первый	>>> 'This is a title'.istitle()
алфавитный символ каждого слова в	False
верхнем регистре, а все остальные	>>> 'Give Me The #\$#@ Ball!'.istitle()
буквенные символы в каждом слове	True
строчные. Возвращает False, если нет	LA DOLL
string.isupper() определяет, являются ли	>>> 'ABC'.isupper()
буквенные символы строки заглавными,	True
возвращает True, если строка s не пустая, и	>>> 'ABC1\$D'.isupper()
все содержащиеся в ней буквенные	True
символы являются заглавными, и в False,	>>> 'Abc1\$D'.isupper()
если нет. Не алфавитные символы	False
игнорируются	

## Вопросы

- ✓ Понятие строки.✓ Как извлечь символ из строки, как извлечь группу символов.✓ Отличие строки от списка.