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 IndexError.

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('Привет, ' + 'Мир!')

* - умножение строк. Значение множителя должно быть целым положительным числом	>>> s = 'py.'
	>>> s * 4
	'py.py.py.py.'

in - оператор принадлежности подстроки, возвращает True, если подстрока входит в строку, и False, если нет.
Есть также оператор not in, у которого обратная логика
	>>> s = 'Python'
	>>> s in 'I love Python.'
	True
	>>> s in 'I love Java.'
	False

Встроенные функции строк	

split() позволяет разбить строку по пробелам. В результате получается список слов.
Может принимать необязательный аргумент-строку, указывающей по какому символу или подстроке следует выполнить разделение
	>>> s = input()
	red blue orange white
	>>> s
	'red blue orange white'
	>>> 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
	>>> '-'.join(sl)
	'red-blue-orange-white'
	
	>>> ''.join(sl)
	'redblueorangewhite'

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
	'red blue orange white'
	>>> 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 другую	>>> letters.replace('DA', 'NET') 'ABCNETCFNET' 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 заданного символа	>>> ord('a')
	97
	>>> ord('#')
	35

chr(n) возвращает символьное значение для данного целого числа.	>>> chr(8364)
	'€'
	>>> chr(8721)
	'∑'

len(s) возвращает длину строки	>>> s = 'Простая строка.'
	>>> len(s)
	15

str(obj) возвращает строковое представление объекта	>>> str(49.2)
	'49.2'
	>>> str(3+4j)
	'(3+4j)'
	>>> str(3 + 29)
	'32'
	>>> str('py')
	'py'

Встроенные методы строк	

string.capitalize() приводит первую букву в верхний регистр, остальные в нижний.	>>> s = 'everyTHing yoU Can IMaGine is rEAl'
	>>> s.capitalize()
	'Everything you can imagine is real'

string.lower() преобразует все буквенные символы в строчные.	>>> 'everyTHing yoU Can IMaGine is rEAl'.lower() 'everything you can imagine is real'

string.swapcase() меняет регистр буквенных символов на противоположный.	>>> 'the sun also rises'.title()
	'The Sun Also Rises'
	
	>>> 'follow us @PYTHON'.title()
	'Follow Us @Python'

string.upper() преобразует все буквенные символы в заглавные.	>>> 'follow us @PYTHON'.upper() 'FOLLOW US @PYTHON'

string.count(<sub>[, <start>[, <end>]])
подсчитывает количество вхождений подстроки в строку.
s.count(<sub>) возвращает количество точных вхождений подстроки <sub> в s:
Количество вхождений изменится, если указать <start> и <end>
	>>> 'foo goo moo'.count('oo')
	3
	>>> 'foo goo moo'.count('oo', 0, 8)
	2

string.endswith(<suffix>[, <start>[, <end>]]) определяет, заканчивается ли строка заданной подстрокой	>>> 'python'.endswith('on')

s.endswith(<suffix>) возвращает, True если s заканчивается указанным <suffix> и False если нет
Сравнение ограничено подстрокой, между <start> и <end>, если они указаны
	True
	>>> 'python'.endswith('or')
	False
	>>> 'python'.endswith('yt', 0, 4)
	True
	>>> 'python'.endswith('yt', 2, 4)
	False

string.find(<sub>[, <start>[, <end>]]) ищет в строке заданную подстроку
s.find(<sub>) возвращает первый индекс в s который соответствует началу строки <sub>
Этот метод возвращает, -1 если указанная подстрока не найдена
Поиск в строке ограничивается подстрокой, между <start> и <end>, если они указаны
	>>> 'Follow Us @Python'.find('Us')
	7
	>>> 'Follow Us @Python'.find('you')
	-1
	>>> 'Follow Us @Python'.find('Us', 4)
	7
	>>> 'Follow Us @Python'.find('Us', 4, 7)
	-1

s.rfind(<sub>) возвращает индекс последнего вхождения подстроки <sub> в s, который соответствует началу <sub>. Как и в .find(), если подстрока не найдена, возвращается -1. Поиск в строке ограничивается подстрокой, между <start> и <end>, если они указаны.	>>> 'Follow Us @Python'.rfind('o') 15

string.isalnum() определяет, состоит ли строка из букв и цифр, возвращает True, если строка s не пустая, а все ее символы буквенно-цифровые (либо буква, либо цифра). В другом случае False	>>> 'abc123'.isalnum()
	True
	>>> 'abc$123'.isalnum()
	False
	>>> ''.isalnum()
	False

string.isalpha() определяет, состоит ли строка только из букв, возвращает True, если строка s не пустая, а все ее символы буквенные. В другом случае False	>>> 'ABCabc'.isalpha()
	True
	>>> 'abc123'.isalpha()
	False

string.isdigit() определяет, состоит ли строка из цифр (проверка на число), возвращает True когда строка s не пустая и все ее символы являются цифрами, а в False если нет	>>> '123'.isdigit()
	True
	>>> '123abc'.isdigit()
	False

string.isidentifier() определяет, является ли строка допустимым идентификатором Python, возвращает True, если s валидный идентификатор (название переменной, функции, класса и т.д.) python, а в False если нет. Вернет True для строки, которая соответствует зарезервированному ключевому слову python, даже если его нельзя использовать	>>> 'foo32'.isidentifier()
	True
	>>> '32foo'.isidentifier()
	False
	>>> 'foo$32'.isidentifier()
	False

string.islower() определяет, являются ли буквенные символы строки строчными, возвращает True, если строка s не пустая, и все содержащиеся в нем буквенные	>>> 'abc'.islower()
	True
	>>> 'abc1$d'.islower()
	True

символы строчные, а False если нет. Не алфавитные символы игнорируются	>>> 'Abc1$D'.islower() False

string.isprintable() определяет, состоит ли строка только из печатаемых символов, возвращает, True если строка s пустая или все буквенные символы которые она содержит можно вывести на экран. Возвращает, False если s содержит хотя бы один специальный символ. Не алфавитные символы игнорируются. Это единственный метод, который возвращает True, если s пустая строка. Все остальные возвращаются False	>>> 'a\tb'.isprintable() # \t - символ табуляции
	False
	>>> 'a b'.isprintable()
	True
	>>> ''.isprintable()
	True
	>>> 'a\nb'.isprintable() # \n - символ перевода строки
	False

string.isspace() определяет, состоит ли строка только из пробельных символов, возвращает True, если s не пустая строка, и все символы являются пробельными, а False, если нет. Наиболее часто встречающиеся пробельные символы — это пробел ' ', табуляция '\t' и новая строка '\n'	>>> ' \t \n '.isspace()
	True
	>>> ' a '.isspace()
	False

string.istitle() определяет, начинаются ли слова строки с заглавной буквы, возвращает True когда s не пустая строка и первый алфавитный символ каждого слова в верхнем регистре, а все остальные буквенные символы в каждом слове строчные. Возвращает False, если нет	>>> 'This Is A Title'.istitle()
	True
	>>> 'This is a title'.istitle()
	False
	>>> 'Give Me The #$#@ Ball!'.istitle()
	True

string.isupper() определяет, являются ли буквенные символы строки заглавными, возвращает True, если строка s не пустая, и все содержащиеся в ней буквенные символы являются заглавными, и в False, если нет. Не алфавитные символы игнорируются	>>> 'ABC'.isupper()
	True
	>>> 'ABC1$D'.isupper()
	True
	>>> 'Abc1$D'.isupper()
	False

Вопросы

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