Строки

- Строка – базовый тип языка Python (string)
- Строки **неизменные** объекты – их нельзя менять, можно только создавать новые строки.

```
>>> 'это строка'
'это строка'
>>> "это тоже строка"
'это тоже строка'
>>> ''#это пустая строка
''
>>> 'использование " в тексте'
'использование " в тексте'
>>> "использование ' в тексте"
"использование ' в тексте"
>>> #или можно использовать экранирование кавычек:
>>> 'использование \' в тексте'
"использование ' в тексте"
>>> 'использование \'@@@\' в тексте'
"использование '@@@' в тексте"
>>>
```

Экранирование символов

- Благодаря экранированию в строках можно использовать символы, вставить которые по другому невозможно
- Экранированный символ предваряется косой чертой \, за которой следует сам символ

\’	Одинарная кавычка (апостроф)
\”	Двойная кавычка
\t	Табуляция
\n	Новая строка (разрыв строки)
****	Обратная косая черта

Экранирование символов

```
>>> print('раз \nдва \nтри')

раз
два
три
>>> print('\tраз \tдва три')

        раз     два три
```

```
>>> print ('I\'m fine')

I'm fine
>>> print ('\tI\'m fine! \nI\'m OK!')

        I'm fine!
I'm OK!
```

Если поместить символ r перед открывающейся кавычкой, строка помечается как необработанная:

```
>>> print(r'C:\newt.txt')
C:\newt.txt
>>> print(r'I'm fine)
SyntaxError: invalid syntax
```

```
>>> print(r'I\'m fine')

I\'m fine
>>>
```

Строки

- Длина строки: Функция *len*(S)
- Строки можно складывать
- Строки можно умножать

```
>>> ss="1+2 = 3"
>>> ss
'1+2 = 3'
>>> len(ss)
7
```

```
>>> s1='наука'
>>> s2='сила'
>>> s1+s2
'наукасила'
>>> s2+s1
'силанаука'
>>> s1*2
'науканаука'
>>> s2*3
'силасиласила'
>>> (s1+s2)*2
'наукасиланаукасила'
>>> s1*2+s2*2
'науканаукасиласила'
>>>
```

Операторы in и not in

- Операторы in и not in применяются к строкам так же, как и к спискам.
- Результатом будет булево значение True или False

'love' **in** 'I love Moscow' >> True

'live' **in** 'I love Moscow' >> False

‘LOve' **in** 'I love Moscow' >> False

'' **in** 'I love Moscow' >> True

‘cats’ not in ‘cats and dogs’ >> False

Комбинирование строк

- Через прибавление +
- Строковая интерполяция %t
- Строковая интерполяция f-строки (от Python 3.6)

```
name_cat = 'Murka'
name_dog = 'Sharik'
age_cat = 7
age_dog = 5

s1='Кошку зовут ' + name_cat + '. Ей ' + str(age_cat) + ' года'

s2='Питомца зовут %s. Возраст %s лет' %(name_cat, age_cat)
s3='Питомца зовут %s. Возраст %s лет' %(name_dog, age_dog)

s4=f'Питомца зовут {name_cat}. Возраст {age_cat} лет'
s5=f'Питомца зовут {name_dog}. Возраст {age_dog} лет'

print(s1)
print(s2)
print(s3)
print(s4)
print(s5)
```

```
Кошку зовут Murka. Ей 7 года
Питомца зовут Murka. Возраст 7 лет
Питомца зовут Sharik. Возраст 5 лет
Питомца зовут Murka. Возраст 7 лет
Питомца зовут Sharik. Возраст 5 лет
```

Срезы

- Срез – извлечение из строки одного символа или фрагмента подстроки или последовательности. Каждый объект, который получается в результате среза – также является строкой типа Str

Срез из одного символа

- S[i] – символ строки S под индексом i;
- нумерация начинается с нуля;
- может быть обратная нумерация

Индекс	[-5]	[-4]	[-3]	[-2]	[-1]	**Пример:**
Строка	с	л	О	в	о	**S[0] = с** **S[1] = л**
Индекс	[0]	[1]	[2]	[3]	[4]	**S[-5] = с** **S[-4] =л**

- При обращении по несуществующему индексу выдается ошибка

Срез с двумя параметрами

- ***Срез с двумя параметрами***
- S[*start:end*] – возвращает срез строки S, начиная с символа под индексом *start* до индекса *end* (*end* не включается);
- Можно использовать как положительные, так и отрицательные индексы;
- Если индексов не существует, ошибка не выдается
- Если не указывать параметр *end*, срез берется до конца строки
- Если не указывать параметр *start*, срез берется до конца строки

Срез с двумя параметрами

S= ‘мама мыла раму’

Строка S	м	а	м	а		м	ы	л	а		р	а	м	у
Индекс	0	1	2	3	4	5	6	7	8	9	10	11	12	13
Индекс	-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1
S[0:4]	м	а	м	а										
S[9:14]											р	а	м	у
S[6:7]							ы							
S[-5:-1]											р	а	м	
S[1:-1]		а	м	а		м	ы	л	а		р	а	м	
S[0:100]	м	а	м	а		м	ы	л	а		р	а	м	у
S[-100:100]	м	а	м	а		м	ы	л	а		р	а	м	у
S[5:]						м	ы	л	а		р	а	м	у
S[:5]	м	а	м	а										

Срез с тремя параметрами

S[*start:end:step*] – возвращает срез строки S, начиная с символа под индексом *start* до индекса *end* (*end* не включается); *step* - шаг

S= ‘мама мыла раму’

Строка S	*м*	*а*	*м*	*а*		*м*	*ы*	*л*	*а*		*р*	*а*	*м*	*у*
Индекс	0	1	2	3	4	5	6	7	8	9	10	11	12	13
Индекс	-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1
S[::2]	*м*		*м*		*м*		*ы*				*р*		*м*	
S[::-1]	*у*	*м*	*а*	*р*		*а*	*л*	*ы*	*м*		*а*	*м*	*а*	*м*
S[1:-1:3]		*а*						*л*			*р*			

Методы работы со строками

- Метод – это функция, применяемая к объекту, в данном случае к строке. Вызывается в виде *Имя_объекта.Имя_метода(параметры)*
- параметры, указанные в [] – необязательны, если они не указаны, берется вся строка
- если не указан параметр [end] – строка берется от параметра [start] до конца строки

Методы работы со строками

S.upper()	Преобразует всю строку к верхнему регистру	S = ’ИнфОрмАтИкА’ S.upper() >>ИНФОРМАТИКА
S.lower()	Преобразует всю строку к нижнему регистру	S = ’ИнфОрмАтИкА’ S.lower() >>информатика
S.isupper()	Возвращает True, если в строке есть хоть одна буква, и вся строка записана в верхнем регистре	S = ‘COLLAPSE’ S.isupper() >> True ‘Collapse’.isupper() >> False
S.islower()	-//- в нижнем регистре	S = ‘collapse’ S.islower() >> True

Методы работы со строками

- Примеры:

1)

```
print('Как настроение?')
feeling=input()
if feeling.lower()=='прекрасно':
    print('Удивительное самообладание!')
else:
    print('Как я вас понимаю...')
```

```
Как настроение?
ПРекрасно
Удивительное самообладание!
```

2)

```
>>> '123'.islower()

False
>>> '123'.isupper()

False
>>> 'abs123'.islower()

True
>>> 'ABC123'.isupper()

True
>>> '123AB1C123'.isupper()

True
```

```
>>> '12ABcd'.upper().lower()

'12abcd'
>>>

>>> 'ABcd'.lower().islower()

True
```

Методы работы со строками

S.isdigit()	Возвращает True, если все символы в строке – цифры	S = ’123’ S1=‘1a2’ S.isdigital() >>True S1.isdigital() >> False
S.isalpha()	Возвращает True, если все символы в строке – буквы	S = ’aBc’ S1=‘1a2’ S.isalpha() >>True S1.isalpha() >> False
S.isalnum()	Возвращает True, если строка непустая, и состоит только из буквенно-цифровых символов	S = ’aBc12’ S1=‘1 a2’ S.isalnum() >>True S1.isalnum() >> False
S.isspace()	Возвращает True, если строка непустая, и состоит только из символов пробела или табуляции	S = ’ ’ S1=‘’ S.isalnum() >>True S1.isalnum() >> False

Методы работы со строками

- Примеры:

1)

```
while True:
    print('Укажите ваш возраст')
    age=input()
    if age.isdigit():
        break
    print('Введите число')
```

```
Укажите ваш возраст
k
Введите число
Укажите ваш возраст
10
```

2)

```
while True:
    print('Введите пароль')
    passw=input()
    if passw.isalnum():
        break
    print('Пароль должен состоятяь из цифр и букв')
```

```
Введите пароль
1a-5
Пароль должен состоятяь из цифр и букв
Введите пароль
fg45
```

Методы работы со строками

метод	описание	пример
S.count(str,[start],[end])	Возвращает количество вхождение подстроки str в строку S. (если вхождений нет, возвращает 0)	S=’123123123’ S.count(‘1’) >> 3 S.count(‘1’,1,6) >> 1
S.find(str,[start],[end])	Возвращает номер первого вхождения подстроки str в строку s (или -1). Важно – возвращает индекс относительно строки, а не относительно среза	S=’программирование и программа’ S.find(‘прог’) >>0 S.find(‘прог’,3) >>19
S.rfind(str,[start],[end])	Возвращает номер последнего вхождения подстроки str в строку s (или -1). Важно – возвращает индекс относительно строки, а не относительно среза	S=’программирование и программа’ S.rfind(‘прог’) >>19 S.rfind(‘прог’,0,5) >>0 S.rfind(‘Прог’,0,5) >>-1
S.replace(шаблон, замена)	Заменяет значение шаблон на значение замена	s='ахахахах' s.replace('а', 'о') >> 'охохохох'

Метод split

- Метод Split разбивает строку на части, используя разделитель, и возвращает эти части списком (слева направо)
- Строка.split([sep]) - возвращает список []
- Sep – разделитель (, * ; и т.д.)
- По умолчанию sep – пробел, при этом крайние пробелы и пустые строки удаляются

Метод split

```
>>> s='1 2 3'

>>> d=s.split()

>>> d

['1', '2', '3']
```

```
>>> s='1 * 2 * 3'
>>> print(s.split('*'))
['1 ', ' 2 ', ' 3']
>>> s='1*2*3'
>>> print(s.split('*'))
['1', '2', '3']
>>>
```

```
>>> s='''Привет!
Не забудь купить молоко.
мама'''
>>> print(s.split('\n'))
['Привет!', 'Не забудь купить молоко.', 'мама']
```

Метод split

```
>>> spisok = input().split()
1 2 3 4 5
>>> spisok
['1', '2', '3', '4', '5']
>>> spisok = input().split(',')
1, 2, 3, 4, 5
>>> spisok
['1', ' 2', ' 3', ' 4', ' 5']
>>> S='red, green, blue'
>>> spisok=S.split(',')
>>> spisok
['red', ' green', ' blue']
>>> S='red green blue'
>>> spisok=S.split()
>>> spisok
['red', 'green', 'blue']
>>> S='red*green*blue'
>>> S
'red*green*blue'
>>> spisok=S.split('*')
>>> spisok
['red', 'green', 'blue']
>>>
```

Метод join

- Метод Join работает в обратную сторону – преобразует список в строку.

sep.join(Объект)

```
>>> spisok
['red', 'green', 'blue']
>>> spisok[1]
'green'
>>> stroka = ','.join(spisok)
>>> stroka
'red,green,blue'
>>> stroka[1]
'e'
>>> stroka = ''.join(spisok)
>>> stroka
'redgreenblue'
>>> stroka = '***'.join(spisok)
>>> stroka
'red***green***blue'
>>>
```

Пример1:

Дан текст:

s = ‘’’Список – последовательность, которая может изменяться и содержать “разнородные” элементы, в том числе и другие списки. В других языках программирования аналогом списков являются массивы, но, как правило, в массивах содержатся однотипные элементы, а в Python такого ограничения нет. Все элементы Python's в списке имеют общее имя и каждый элемент имеет свой собственный индекс (порядковый номер). Иными словами, список (list) – структура данных для хранения объектов различных типов. Элементы списка заключаются в квадратные скобки [] и отделяются друг от друга запятой. Нумерация элементов в списке начинается с 0.’’’

Длина текста: len(s) 616

Пример1:

- Найти, сколько раз в тексте встречается слово ' список ' во всех возможных формах

s.lower().count('спис')

 7

- Вывести на экран количество предложений в тексте

s.lower().count('.')

 6

Пример1:

- Вывести на экран второе предложение, записанное прописными буквами.

```
>>> s.find('.')
117
>>> s.find('.',118)
280
>>> print (s.upper()[118:280])
```

```
 В ДРУГИХ ЯЗЫКАХ ПРОГРАММИРОВАНИЯ АНАЛОГОМ СПИСКОВ ЯВЛЯЮТСЯ МАССИВЫ, НО, КАК ПРАВИЛО, В МАССИВАХ СОДЕРЖАТСЯ ОДНОТИПНЫЕ ЭЛЕМЕНТЫ, А
В PYTHON ТАКОГО ОГРАНИЧЕНИЯ НЕТ
```

Или:

```
>>> d = s.split('.')
>>>print( d[1].upper() )
```

Пример1:

- Вывести на экран последнее предложение, заменив цифры на символы.

- >>> s.rfind('.',0,615)
- 570
- >>> print (s.replace('0','нуля')[571:])

Нумерация элементов в списке начинается с нуля.

Или:

>>> d = s.split('.')

>>>print(d[5].replace(‘0’, ‘нуля’))

Пример2:

Даны строки, которые получены 25 января. Каждая строка содержит:

- код зачетки студента
- предмет, по которому студент сдавал экзамен
- оценка за экзамен
- Элементы строки разделены символом *

- S1= 'kod700*физика*4*'
- S2= 'kod706*экономика*3*'
- S3= 'kod702*философия*5*'
- S4= 'kod708*Физика*5*'
- S5= 'kod709*Физика*4*'
- S6= 'kod710*Экономика*3*'
- S7= 'kod716*Экономика*3*'
- S8= 'kod717*Физика*4*'
- S9= 'kod718*экономика*5*'
- S10= 'kod719*физика*5*'

Пример2:

1) Найти: количество студентов, сдававших экзамен по каждому предмету

Все строки соберем в единую строку

s=S1+S2+S3+S4+S5+S6+S7+S8+S9+S10

Строку запишем в массив

m1=s.split('*')

```
print ('физика',s.lower().count('физика'))
print ('экономика',s.lower().count('экономика'))
print ('философия',s.lower().count('философия'))
```

Пример2:

2) Вывести на экран количество 2, если их нет, вывести сообщение: все студенты сдали экзамен удовлетворительно

```
count2=0
for i in range(2,len(m1),3):
   if  m1[i]=='2': count2+=1
if count2==0:
    print('все студенты сдали экзамены удовлетворительно')
else:
    print('двоек =' ,count2)
```

```
stemp=''
for i in range(2,len(m1),3):
    stemp+=m1[i]
if stemp.count('2')==0:
    print ('no')
else:
    print ('всего двоек', stemp.count('2'))
```

Пример2:

1) Вывести коды зачеток студента, получивших 5 по физике
2) Вывести коды зачеток студентов, получивших 3 по экономике
3) Вывести средний балл всех студентов
4) Вывести на экран массив построчно: код, предмет, оценка, где вместо 5 будет записано ‘отлично’

Пример2:

```
summ=0
kol=0
snew=[]  #новый список для всех элементов
skod=[]  #список для кодов зачеток
spr=[]   #список для наименований предметов
smark=[] #список для оценок
#Все элементы строки записываем в список
snew=s.lower().split('*')
N=len(snew)
#Создание списков для кодов, предметов, оценок
for i in range(0,N-2,3):
    skod.append(snew[i])
    spr.append(snew[i+1])
    smark.append(snew[i+2])
```

Пример2:

```
for i in range(len(skod)):
#Задание 1
    if smark[i]=='5' and spr[i]=='физика':
        print (skod[i], spr[i],smark[i])
#Задание 2
    if smark[i]=='3' and spr[i]=='экономика':
        print (skod[i], spr[i],smark[i])
#Задание 3
    summ+=int(smark[i])
    kol=kol+1
print ('sr=',summ/kol)
#Задание 4
for i in range(len(skod)):
    print (skod[i], spr[i],smark[i].replace('5','отлично'))
```

```
физика 5
двоек = 1
всего двоек 1
kod706 экономика 3
kod708 физика 5
kod710 экономика 3
kod719 физика 5
sr= 4.0
```

```
kod700 физика 4
kod706 экономика 3
kod702 философия отлично
kod708 физика отлично
kod709 физика 4
kod710 экономика 3
kod716 экономика 2
kod717 физика 4
kod718 экономика отлично
kod719 физика отлично
```