**Строки (str)**

Этот тип данных часто встречается, потому что люди работают не только с числами, но и с текстовыми данными.

Строка это последовательность символов.

Считывание строки s=input()

Вы можете присваивать строке некоторый текст

s="Hello world" или s='Hello'

Основной смысл использования “ и ‘ для того, чтобы внутри строки можно было использовать кавычки или апострофы

s='He said "Hello!"'

Можно обойтись и одними s="He said \"Hello!\""

В тройных апострофах или тройных кавычках (в этом случае строка может содержать символы конца строки):

'''spam''', """spam

new line"""

Строки могут содержать экранированные последовательности (\t – табуляция, \n – переход на новую строку):

"s\tp\nam"

Если в строке нужно вывести обратную косую черту, то ее нужно повторить дважды:

'Symbol \\ can be also used.'

Неформатированные строки (\символ не воспринимается как управляющая последовательность):

r"C:\new\test\0012"

**Пустая строка**

s = ""

s = str()

имя типа – это конструктор объекта. Если конструктор задан без параметров, то создается пустой объект.

**Конкатенация строк**

s = "a"

t = "b"

print(s + t)

**Умножение строки на число**  – строка повторяется указанное количество раз

s = "ab"

t = s \* 3 # s+s+s=”ababab”

**Переход на новую строку**

s = "Hello \n world"

s = "\\n" сама последовательность \n

**Сравнение строк**

Строки сравниваются в лексикографическом порядке/

**Длина строки**

len(s)

**Проверка, что подстрока s2 содержится в s**

s2 **in** s

'm' **in** 'team'

s2 **not in** s то же, что not (s2 in s)

'I' **not in** 'team'

**Обращение к символам строки**

Все символы в строке пронумерованы от 0 до len(s)-1. Можно обращаться к символу по его номеру. Если обратиться к несуществующему символу s[100], если в строке 3 символа, то возникнет ошибка Index error.

Можно обращаться к символам строки с конца используя отрицательную индексацию.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| H | e | l | l | o |  | w | o | r | l | d | ! |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| -12 | -11 | -10 | -9 | -8 | -7 | -6 | -5 | -4 | -3 | -2 | -1 |

s[-2] второй с конца символ

s[-len(s)] первый символ.

Разность между положительным и отрицательным индексом всегда равна длине строки. 11-(-1)=0(-12)=12

С отрицательными индексами можно обращаться и к элементам списка.

Можно и перебрать символы строки в цикле for:

**for** c **in** line:

**print**(c)

**В Python строка неизменяемый объект!**

Вы не можете исправить в строке какой-то символ. Нужно создавать новую строку. Для этого используют срезы.

**Срезы**

Срез (slice) это основной инструмент для работы со строками.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| s=” | a | b | c | d | e | f | g | h | i | j | k | l | “ |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |  |

1. **s[a:b]** из строки s берутся символы с индексами от a до b, не включая b.

Длина строки b-a. Индексацию можно использовать как положительную, так и отрицательную.

s[3:7] # defg

s[5:1] пустая строка

s[1:-1] строка без первого и последнего символа

В срезе можно опускать первую или последнюю часть

s[:4] # abcd

s[:n] первые n символов

s[5:] # fghijkl с 5 символа и до конца

s[-5:] последние 5 символов

s[:] вся строка

**2. s[a:b:d]** из строки берутся символы, начиная с a до b (не включая b) с шагом d

Длина строки (b-a)//d. Могут отсутствовать некоторые части.

s[3:9:2] #dfh

s[3:10:2] #dfhj

s[::2] символы на четных местах

Если d<0, то срез разворачивается

s[9:2:-2] # jhfd

s[9:2:-1] # jihgfed

s[::-1] разворот строки в обратном порядке

Срезы можно комбинировать

Отбросить крайние символы и развернуть s[-2:0:-1] или s[1:-1][::-1]

Срезы можно брать и у списков. Срезы тоже будут являться копией.

**Методы строк**

s="abracadabra"

**1. s.find(t)** индекс первого вхождения строки t в строку s. Если строка t не входит в s, то возвращается -1.

s.find("ac")=3

s.find("aa")=-1

**s.rfind(t)** индекс последнего вхождения строки t в строку s. Если строка t не входит в s, то возвращается -1.

s.rfind("ab")=7

**s.find(t,a,b)** индекс последнего вхождения строки t в строку s среди символов среза s[a:b]. Если строка t не входит в s, то возвращается -1.

**2. s.replase(t1,t2**) меняет все вхождения строки t1 на строку t2. Метод строку s не изменяет, он возвращает новую строку.

t=s.replase("a","aa") удвоить все буквы a

t=s.replase("a","aa",k) удвоить все буквы a, выполнив не более k замен.

**3.** **s.count(t)** подсчитывает сколько в строку s входит строка t. Считаются непересекающиеся вхождения.

s = "aaaa"

s.count("aa") = 2

Если в строке слова разделены одним пробелом, то количество слов в строке можно найти s.count(" ") + 1

**список символов строки s**

list('Привет') == ['П', 'р', 'и', 'в', 'е', 'т']

**получить какое-то представление в виде строки любого объекта x**

str(25) == '25'

Обратите внимание, что **никакие** методы строки, включая s.replace(...), не изменяют саму строку s. Все они лишь возвращают изменённую строку

**Код символа** **k = ord(c)**

**Символ по коду c = chr(k)**

**Проверить является ли символ цифрой**

1 вариант

ord('0') <= ord(c) <= ord('9')

2 вариант

'0' <= c <= '9'

3 вариант

c.**isdigit**()

**Проверить является ли символ строчной латинской буквой**

1 вариант

ord('a') <= ord(c) <= ord('z')

2 вариант

'a' <= c <= 'z'

**Проверить является ли символ заглавной латинской буквой**

1 вариант

ord('A') <= ord(c) <= ord('Z')

2 вариант

'A' <= c <= 'Z'

**Проверить является ли символ буквой (в том числе и кириллической)**

c.**isalpha**()

**Проверить является ли символ буквой или цифрой**

c**.isalnum**()

**Строчную латинскую букву сделать заглавной**

1 вариант

**if** "a" <= c <= "z":

c =chr(ord(c) - 32)

2 вариант (в том числе и кириллическую)

c = с.**upper**()

**Заглавную латинскую букву сделать строчной**

1 вариант

**if** "A" <= c <= "Z":

c =chr(ord(c) + 32)

2 вариант (в том числе и кириллическую)

c = с.**lower**()

**Сделать строку, состоящую из строчных букв, начинающуюся с заглавной**

s = s.**capitalize**()

**Удаление разделителей**

s.lstrip()

s.rstrip()

s.strip() – строка s, у которой удалены символы пустого пространства (пробелы, табуляции) в начале (в конце / с обеих сторон)

' Привет! '.strip() == 'Привет!'

Бывает удобно строить цепочки вызовов методов: **s.strip().lower().replace('ё', 'е')** выдаст строку s, в которой убраны начальные и конечные пробелы, все буквы сделаны маленькими, после чего убраны все точки над «ё». Так, например, строка '  Зелёный  клён    ' превратится в 'зеленый  клен'.

**Форматирование строк**

print('{}({}, {}, {})'.format(name, x, y, z)) #координаты точки

species, meal, name = 'мумзик', 'яблоки', 'Вовочка'

print('Ни один {0} не ест {1}. {2} ест {1}, значит {2} не {0}.'.format(species, meal, name))