Python.

Строки.

Строка в языке Python — это последовательность символов, заключённая в апострофы или двойные кавычки.

```
s1 = 'Привет'
s2 = "дорогой друг"
```

Строки можно складывать. Эта операция называется конкатенация, или сложение, или слияние строк и обозначается знаком плюс.

Например, при выполнении следующей программы

```
s1 = 'Привет'
s2 = "дорогой друг"
s = s1 + ', ' + s2 + "!"
print(s)
```

будет напечатано

```
Привет, дорогой друг!
```

Строки можно сравнивать.

Сравнение осуществляется посимвольно слева направо в соответствии с тем порядком, в котором символы хранятся в компьютере. Латинские и русские буквы хранятся в алфавитном порядке, цифры – от 0 до 9 в порядке возрастания.

Пример:

‘AB’ > ‘AA’
‘A’ < ‘AB’
‘DC’ > ‘ABCDE’
‘ABCE’ > ‘ABCD’
“123” < “2”
‘A’ < ‘a’
“Иванов” < “Петров”

Часто бывает необходимо обратиться не ко всей строке, а к её части. В языке Python для этого используют ***срезы***.

Простейшая ситуация – это когда нужно взять один конкретный символ из строки. В этом случае в квадратных скобках после имени строки указывается номер порядковый символа (нумерация начинается с нуля).

`string[k]`

Здесь **`string`** — это имя строковой переменной, а **`k`** — номер символа в этой строке.

Например, в результате выполнения следующей программы

```
s = 'Привет!'
print(s[0])
print(s[2])
print(s[6])
print(s[-1])
print(s[-2])
```

будет напечатано

```
П
и
!
!
т
```

Обратим внимание на то, что если индекс (номер символа) отрицательный, то отсчет ведется с конца строки.

Если индекс выходит за границы строки (например, для приведенной выше строки обратиться к s[10]), то при выполнении программы произойдет ошибка.

IndexError: string index out of range

Чаще бывает необходимо взять из строки не один символ, а несколько.

s = string[a:b]

В этом случае в переменную **s** будет записана часть строки **string** начиная с символа с номером **a** и заканчивая символом с номером **b-1.**

Можно пропустить одну из границ **a** или **b**.

s1 = string[:b]
s2 = string[a:]

Тогда в переменной **s1** будет часть строки **string** начиная с самого первого символа и до символа с номером **b-1,** а в переменной **s2** будет часть строки **string** начиная с символа с номером **a** и до конца строки.

Если опустить обе границы, то вернется вся строка.

s = string[:]

Например, в результате выполнения следующей программы

```
s = 'Привет!'
print(s[1:4])
print(s[:2])
print(s[3:])
print(s[:])
print(s[2:-2])
```

будет напечатано

```
рив
Пр
вет!
Привет!
иве
```

При создании среза может быть использован и третий параметр.

s = string[a:b:c]

В этом случае в переменную **s** будет записана символы из строки **string** начиная с символа с номером **a** и заканчивая символом с номером **b-1,** с шагом **c.**

Здесь также могут быть пропущены как параметр **a** , так и параметр **b** , а также оба этих параметра.

Например, в результате выполнения следующей программы

```
s = 'Привет, дорогой друг!'
print(s[1:10:2])
print(s[:10:3])
print(s[10::3])
print(s[::4])
print(s[::-1])
```

будет напечатано

```
рвт о
Пв,о
родг
Педгд!
!гурд йогород ,тевирП
```

Обратим внимание на последний вывод. Если задать шаг равный ***-1***, то получим перевернутую строку.

Часто бывает необходимо проверить является ли символ или набор символов частью какой-то строки. Для этого удобно пользоваться оператором ***in***.

Например, для проверки является ли буква гласной русской, можно написать следующую функцию.

```
def glas(s):
    return s in 'аоеёиыуюэя'
```

Она будет возвращать ***True***, если в параметре ***s*** гласная буква, и ***False*** в противном случае.

Для перебора символов в строке удобно использовать циклы. Напишем фрагмент программы, которая будет считать количество гласных русских букв в строке.

```
def glas(s):
    return s in 'аоеёиыуюэя'

s = 'Привет, дорогой друг!'
k = 0
for i in s:
    if glas(i):
        k+=1
print(k)
```

В результате работы программы будет выведено число 6.

Для работы со строками существует ряд стандартных функций. Рассмотрим их.

Для определения длины строки (количества символов в строке) служит функция ***len(s).***

Предыдущий пример можно переписать следующим образом.

```
def glas(s):
    return s in 'аоеёиыуюэя'

s = 'Привет, дорогой друг!'
k = 0
for i in range(len(s)):
    if glas(s[i]):
        k+=1
print(k)
```

Для работы со строками можно использовать функции max() и min() . Они вернут, соответственно, символы с максимальным и минимальным кодом.

```
s = 'dfbdfagntyd'
print(max(s))
print(min(s))
```

В результате работы данной программы будет выведено

```
y
a
```

Для того чтобы узнать код символа можно воспользоваться функцией ***ord()***. Для получения символа по его коду используется функция ***chr()***.

Например, в результате работы следующей программы

```
print(ord('A'))
print(chr(66))
```

будет напечатано

```
65
B
```

Используя эти функции, можно легко написать программу, которая выведет английский алфавит.

```
alph = ''
for i in range(26):
    alph+=chr(ord('A')+i)
print(alph)
```

В результате будет напечатано

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

Кроме функций, для работы со строками используется ряд методов. Некоторые из них приведены в следующей таблице.

Операция	**Описание**	**Пример**
s2 **in** s	Проверка, что подстрока s2 содержится в s	'm' in 'team'
s2 **not in** s	Проверка, что подстрока s2 не содержится в s то же, что not (s2 in s)	'I' not in 'team'
s.find(s2) s.rfind(s2)	Индекс начала первого или последнего вхождения подстроки s2 в s (вернет -1, если s2 not in s)	s = 'abracadabra' s.find('ab') == 0 s.rfind('ab') == 7 s.find('x') == -1
s.count(s2)	Количество неперекрывающихся вхождений s2 в s	'abracadabra'.count('a') == 5
s.startswith(s2) s.endswith(s2)	Проверка, что s начинается с s2 или оканчивается на s2	'abracadabra'.startswith('abra')
s.isdigit() s.isalpha() s.isalnum()	Проверка, что в строке s все символы — цифры, буквы (включая кириллические), цифры или буквы соответственно	'100'.isdigit() 'abc'.isalpha() 'E315'.isalnum()
s.islower() s.isupper()	Проверка, что в строке не встречаются большие буквы, маленькие буквы. Обратите внимание, что для обеих этих функций знаки препинания и цифры дают True	'hello!'.islower() '123PYTHON'.isupper()

s.lower() s.upper()	Строка s, в которой все буквы (включая кириллические) приведены к верхнему или нижнему регистру, т. е. заменены на строчные (маленькие) или заглавные (большие)	'Привет!'.lower() == 'привет!' 'Привет!'.upper() == 'ПРИВЕТ!'
s.capitalize()	Строка s, в которой первая буква — заглавная	'привЕТ, ДРУГ!'.capitalize() == 'Привет, друг!'
s.title()	Строка s, в которой первая буква каждого слова— заглавная	'привЕТ, ДРУГ!'.capitalize() == 'Привет, Друг!'
s.lstrip() s.rstrip() s.strip()	Строка s, у которой удалены символы пустого пространства (пробелы, табуляции) в начале, в конце или с обеих сторон	' Привет! '.strip() == 'Привет!'
s.replace(s2, s3)	Строка s, в которой все неперекрывающиеся вхождения s2 заменены на s3 Есть необязательный третий параметр, с помощью которого можно указать, сколько раз производить замену	'Раз два три!'.replace('а', 'я') =='Ряз двя три!' 'Раз два три!'.replace('а', 'я', 1) == 'Ряз два три!'