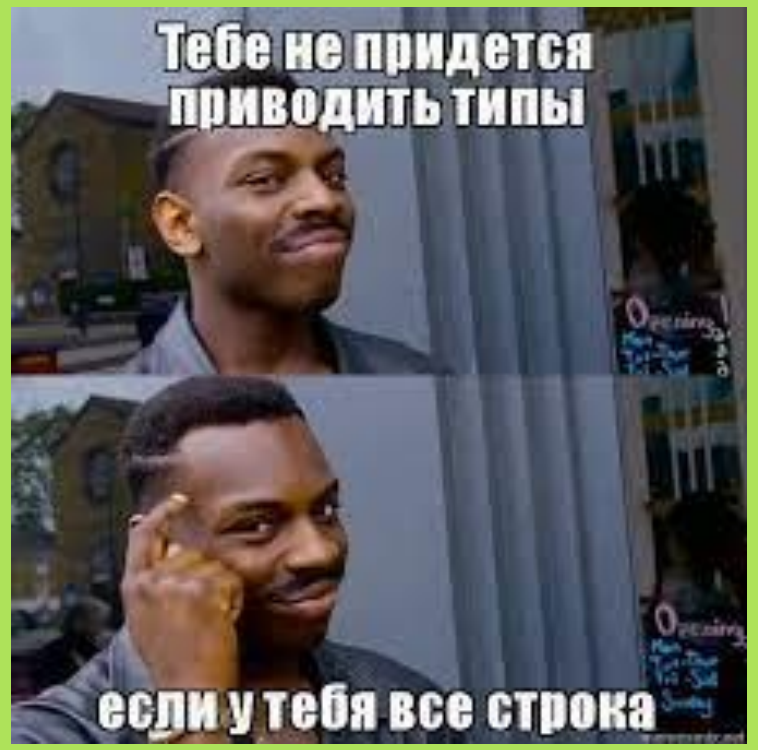
Введение в программирование на Python

Куляшов Михаил, м.н.с. НТУ «Сириус»
Telegram: @mishakula

Строки

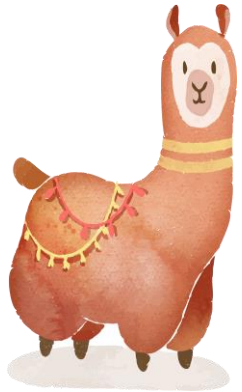


Типы данных по изменяемости

Boolean (логические типы данных)

Numbers (числа)

String (строки)



Изменяемые

Неизменяемые

List (списки)

Set (множества)

Tuples (кортежи)

Dictionaries (словари)

Типы данных по упорядоченности

String (строки)

Упорядоченные

Неупорядоченные

List (списки)

Set (множества)

Tuples (кортежи)

Dictionaries (словари)

Пример строки

In [67]:
```
s = 'lesson 1'
s
```

Out [67]:
```
'lesson 1'
```

In [68]:
```
# Посмотрим на типы данных переменных s и переменной n
s = '1'
n = 1
print(type(s))
print(type(n))
```

Out [68]:
```
<class 'str'>
<class 'int'>
```

In [70]:
```
# Результат сложения двух строк
new_s = s + s
new_s
```

Out [70]:
```
'11'
```

In [72]:
```
#Сложение 3 строк
s1 = 'строка номер 1'
s2 = 'новая строка номер 2'
s3 = 'последняя строка номер 3'

result_s = s1 + s2 + s3
result_s
```

Операции со строками

Оператор	Значение
`+ или +=`	Сложение
`==`	Сравнение
`* Или *=`	Умножение (дублирование)
`int(), list() и т.п.`	Преобразование к другим типам данных

Примеры операции со строками

In [69]:

```
# Преобразуем число 1 записанное в строке, в тип данных integer и float
s = '1'
print('type of s = ', type(s))
int_s = int(s)
print('type of int_s = ', type(int_s))
float_s = float(s)
print('type of float_s = ', type(float_s))

#По аналогии с числовыми типами данных используем str() для преобразования в строку

back_to_string_int = str(int_s)
print('type of back_to_string_int = ', type(back_to_string_int))
back_to_string_float = str(float_s)
print('type of back_to_string_float = ', type(back_to_string_float))
```

Примеры операции со строками

In [72]:
```
#Сложение 3 строк
s1 = 'строка номер 1'
s2 = 'новая строка номер 2'
s3 = 'последняя строка номер 3'

result_s = s1 + s2 + s3
result_s
```

Out [72]:
```
'строка номер 1новая строка номер 2последняя строка номер 3'
```

In [73]:
```
# Также строки можно сравнить между сбой
s1 == s2
```

Out [73]:
```
False
```

In [77]:
```
# Сокращенное сложение строк. Т.к. на предыдуще
s1 = 'строка номер 1'
s1 += s2
s1
```

Out [77]:
```
'строка номер 1новая строка номер 2'
```

In [79]:
```
#Утроим строку
s4 = 'spaming'
s4 *= 3
s4
```

Out [79]:
```
'spamingspamingspaming'
```

Синтаксические конструкции используемые в строках

Символ	Значение
\\’	Одинарная кавычка
\”	Двойная кавычка
\t	Табуляция
\n	Перенос строки
\r	Перенос курсора
\f	Прогон страницы
\b	Backspace
\v	Вертикальная табуляция

Примеры использования синтаксических конструкций

In [78]:

```
# Разделим строку с помощью пробела, таблуляции и запятой. Также еще раз создадим переменные s1,s2,s3.

s1 = 'строка номер 1'
s2 = 'новая строка номер 2'
s3 = 'последняя строка номер 3'

space_s = s1 + ' ' + s2 + ' ' + s3
tsv_s = s1 + '\t' + s2 + '\t' + s3
csv_s = s1 + ',' + s2 + ',' + s3

print(tsv_s)
print(csv_s)
```

Out [78]:

```
строка номер 1  новая строка номер 2    последняя строка номер 3
строка номер 1,новая строка номер 2,последняя строка номер 3
```

Индексация и срезы

Строка	A	L	P	A	C	A	S
Индекс символа	0	1	2	3	4	5	6
Обратный индекс	-7	-6	-5	-4	-3	-2	-1

Примеры работы с индексами строк

In [80]:
```
# Получим букву p из строки alpacas
s5 = 'alpacas'
s5[2]
```

Out [80]:
```
'p'
```

In [81]:
```
# Получим последний символ строки с использованием прямого индекса (т.е. отсчет от начала слова)
s5[6]
```

Out [81]:
```
's'
```

In [82]:
```
# Получим последний символ строки с использованием обратного индекса
s5[-1]
```

Out [82]:
```
's'
```

In [83]:
```
len(s5)
```

Out [83]:
```
7
```

Примеры работы с индексами строк

In [80]:
```
# Получим букву p из строки alpacas
s5 = 'alpacas'
s5[2]
```

Out [80]:
```
'p'
```

In [81]:
```
# Получим последний символ строки с использованием прямого индекса (т.е. отсчет от начала слова)
s5[6]
```

Out [81]:
```
's'
```

In [82]:
```
# Получим последний символ строки с использованием обратного индекса
s5[-1]
```

Out [82]:
```
's'
```

In [83]:
```
len(s5)
```

Out [83]:
```
7
```

Примеры работы с индексами строк

In [80]:
```
# Получим букву p из строки alpacas
s5 = 'alpacas'
s5[2]
```

Out [80]:
```
'p'
```

In [81]:
```
# Получим последний символ строки с использованием прямого индекса (т.е. отсчет от начала слова)
s5[6]
```

Out [81]:
```
's'
```

In [82]:
```
# Получим последний символ строки с использованием обратного индекса
s5[-1]
```

Out [82]:
```
's'
```

In [83]:
```
len(s5)
```

Out [83]:
```
7
```

Срезы строк

Строка	A	L	P	A	C	A	S
Индекс символа	0	1	2	3	4	5	6
Обратный индекс	-7	-6	-5	-4	-3	-2	-1

Примеры срезов строк

In [84]:
```
#Срез строки s5 от 2 до последнего элемента
s5[1:]
```

Out [84]:
```
'lpacas'
```

In [85]:
```
#Срез строки s5 от 2 до 4 элементо
s5[1:4]
```

Out [85]:
```
'lpa'
```

In [86]:
```
#Срез строки s5 от 2 до -1 элементо
s5[1:-1]
```

Out [86]:
```
'lpaca'
```

In [87]:
```
#Использование шага 2 при взятии среза
s5[::2]
```

Out [87]:
```
'apcs'
```

Примеры срезов строк с использованием шага

In [87]:
```
#Использование шага 2 при взятии среза
s5[::2]
```

Out [87]:
```
'apcs'
```

In [88]:
```
#Использование шага 2 при взятии среза не от начала, а от 2 элемента
s5[1::2]
```

Out [88]:
```
'laa'
```

In [89]:
```
#Использование среза и шага -1 для получение строки в обратном порядке
s5[::-1]
```

Out [89]:
```
'sacapla'
```

Основные методы строк

Метод	Описание
strip()	Удаление пробельных символов в начале и в конце строки
split()	Разделяет строку по заданному символу
replace()	Заменяет заданное количество символов в строке на новые, указанные в методе
join()	Метод преобразования списка в строку
count()	Посчитать количество символов в строке
find()	Найти первое вхождение символа в строке (либо слева, либо справа)
rfind()	

Основные методы строк: strip

Исходная строка:

```
' GCTGGGGGGCTATAAAAAGAGGAGGCGCAGCGGAGAGACAGAGACGGACAAAGACCAAGAGAGCGAAACC
AGAGAGGAGCAAGAAGCAGCAAACACCAGACCATTCTTTGACCGACTCCAGCATGGGCTCCTCCGCCATC '
```

Строка после использования метода strip():

```
'GCTGGGGGGCTATAAAAAGAGGAGGCGCAGCGGAGAGACAGAGACGGACAAAGACCAAGAGAGCGAAACC
AGAGAGGAGCAAGAAGCAGCAAACACCAGACCATTCTTTGACCGACTCCAGCATGGGCTCCTCCGCCATC'
```

Основные методы строк: replace

Исходная строка:

```
'GCTGGGGGGCTATAAAAAGAGGAGGCGCAGCGGAGAGACAGAGACGGACAAAGACCAAGAGAGCGAAACC
AGAGAGGAGCAAGAAGCAGCAAACACCAGACCATTCTTTGACCGACTCCAGCATGGGCTCCTCCGCCATC'
```

Строка после использования метода replace(' ', ''):

```
'GCTGGGGGGCTATAAAAAGAGGAGGCGCAGCGGAGAGACAGAGACGGACAAAGACCAAGAGAGCGAAACCAGAGAGGAGCAAGAAGCAGCAA
ACACCAGACCATTCTTTGACCGACTCCAGCATGGGCTCCTCCGCCATC'
```

Основные методы строк: replace

Исходная строка:

```
'GCTGGGGGGCTATAAAAAGAGGAGGCGCAGCGGAGAGACAGAGACGGACAAAGACCAAGAGAGCGAAACCAGAGAGGAGCAAGAAGCAGCAA
ACACCAGACCATTCTTTGACCGACTCCAGCATGGGCTCCTCCGCCATC'
```

Строка после использования метода replace(‘T', ‘U’, 1):

```
'GCUGGGGGGCTATAAAAAGAGGAGGCGCAGCGGAGAGACAGAGACGGACAAAGACCAAGAGAGCGAAACCAGAGAGGAGCAAGAAGCAGCAA
ACACCAGACCATTCTTTGACCGACTCCAGCATGGGCTCCTCCGCCATC'
```

Основные методы строк: replace

Исходная строка:

```
'GCTGGGGGGCTATAAAAAGAGGAGGCGCAGCGGAGAGACAGAGACGGACAAAGACCAAGAGAGCGAAACCAGAGAGGAGCAAGAAGCAGCAA
ACACCAGACCATTCTTTGACCGACTCCAGCATGGGCTCCTCCGCCATC'
```

Строка после использования метода replace(‘T', ‘U’):

```
'GCUGGGGGGCUAUAAAAAGAGGAGGCGCAGCGGAGAGACAGAGACGGACAAAGACCAAGAGAGCGAAACCAGAGAGGAGCAAGAAGCAGCAA
ACACCAGACCAUUCUUUGACCGACUCCAGCAUGGGCUCCUCCGCCAUC'
```

Основные методы строк: split

Исходная строка:

```
'GCTGGGGGGCTATAAAAAGAGGAGGCGCAGCGGAGAGACAGAGACGGACAAAGACCAAGAGAGCGAAACC
AGAGAGGAGCAAGAAGCAGCAAACACCAGACCATTCTTTGACCGACTCCAGCATGGGCTCCTCCGCCATC'
```

Строка после использования метода split(' '):

```
['GCTGGGGGGCTATAAAAAGAGGAGGCGCAGCGGAGAGACAGAGACGGACAAAGACCAAGAGAGCGAAACC',
'AGAGAGGAGCAAGAAGCAGCAAACACCAGACCATTCTTTGACCGACTCCAGCATGGGCTCCTCCGCCATC']
```

Основные методы строк: split

Исходная строка:

```
'GCTGGGGGGCTATAAAAAGAGGAGGCGCAGCGGAGAGACAGAGACGGACAAAGACCAAGAGAGCGAAACC
AGAGAGGAGCAAGAAGCAGCAAACACCAGACCATTCTTTGACCGACTCCAGCATGGGCTCCTCCGCCATC'
```

Строка после использования метода split(' '):

```
['GCTGGGGGGCTATAAAAAGAGGAGGCGCAGCGGAGAGACAGAGACGGACAAAGACCAAGAGAGCGAAACC',
'AGAGAGGAGCAAGAAGCAGCAAACACCAGACCATTCTTTGACCGACTCCAGCATGGGCTCCTCCGCCATC']
```

Основные методы строк: replace + split

Исходная строка:

```
'GCTGGGGGGCTATAAAAAGAGGAGGCGCAGCGGAGAGACAGAGACGGACAAAGACCAAGAGAGCGAAACC
AGAGAGGAGCAAGAAGCAGCAAACACCAGACCATTCTTTGACCGACTCCAGCATGGGCTCCTCCGCCATC'
```

После использования методов replace(' ', '').split('CC'):

```
['GCTGGGGGGCTATAAAAAGAGGAGGCGCAGCGGAGAGACAGAGACGGACAAAGA’,
'AAGAGAGCGAAA’,
'AGAGAGGAGCAAGAAGCAGCAAACA’,
'AGA’,
'ATTCTTTGA’,
'GACT’,
'AGCATGGGCT’,
'T’,
'G’,
 'ATC']
```

Основные методы строк: join

Исходный список строк:

```
['GCTGGGGGGCTATAAAAAGAGGAGGCGCAGCGGAGAGACAGAGACGGACAAAGA',
'AAGAGAGCGAAA',
'AGAGAGGAGCAAGAAGCAGCAAACA',
'AGA',
'ATTCTTTGA',
'GACT',
'AGCATGGGCT',
'T',
'G',
 'ATC']
```

После использования метода ‘CC’.join():

```
'GCTGGGGGGCTATAAAAAGAGGAGGCGCAGCGGAGAGACAGAGACGGACAAAGACCAAGAGAGCGAAACCAGAGAGGAGCAAGAAGCAGCAA
ACACCAGACCATTCTTTGACCGACTCCAGCATGGGCTCCTCCGCCATC'
```

Основные методы строк: find и rfind

Исходная строка:

```
'GCTGGGGGGCTATAAAAAGAGGAGGCGCAGCGGAGAGACAGAGACGGACAAAGACCAAGAGAGCGAAACCAGAGAGGAGCAAGAAGCAGCAA
ACACCAGACCATTCTTTGACCGACTCCAGCATGGGCTCCTCCGCCATC'
```

In [13]:

```
#Поиск набора нуклеотидов CG в последовательности
print(return_to_string.find('CG'))
print(return_to_string.rfind('CG'))
```

Out [13]:

```
25
133
```

In [101]:

```
#Посмотрим действительно ли там есть CG последовательности
print(return_to_string[25:40])
print(return_to_string[133:])
```

Out [101]:

```
CGCAGCGGAGAGACA
CGCCATC
```

Основные методы строк

Метод	Что делает
islower()	Состоит ли строка из символов в нижнем регистре
isupper()	Состоит ли строка из символов в верхнем регистре
isalnum()	Состоит ли строка из цифр или букв
isalpha()	Состоит ли строка из букв
isdigit()	Состоит ли строка из цифр
isspace()	Состоит ли строка из неотображаемых символов (символы табуляции, переноса строки, и т.п.)
istitle()	Начинаются ли слова в строке с заглавной буквы
startswith()	Начинается ли строка с заданного шаблона
endswith()	Заканчивается ли строка заданным шаблоном

Основные методы строк

Исходная строка:

```
'GCTGGGGGGCTATAAAAAGAGGAGGCGCAGCGGAGAGACAGAGACGGACAAAGACCAAGAGAGCGAAACCAGAGAGGAGCAAGAAGCAGCAA
ACACCAGACCATTCTTTGACCGACTCCAGCATGGGCTCCTCCGCCATC'
```

Основные методы строк

In [104]:
```
#Посмотрим, состоит ли наша строка из букв в нижнем регистре
return_to_string.islower()
```

Out [104]:
```
False
```

In [105]:
```
#Посмотрим, есть ли цифры в нашей строке, т.е. состоит ли она только из букв
return_to_string.isalpha()
```

Out [105]:
```
True
```

In [106]:
```
#Проверим начинается ли наша строка с последовательности GCTGGG
return_to_string.startswith('GCTGGG')
```

Out [106]:
```
True
```

In [107]:
```
# А теперь проверим, заканчивается ли она последовательностью GCTGGG ?
return_to_string.endswith('GCTGGG')
```

Out [107]:
```
False
```

Основные методы строк

Метод	Что делает
capitalize()	Переводит первый символ строки в верхний регистр, а все остальные в нижний
swapcase()	Переводит символы нижнего регистра в верхний, а верхнего – в нижний
title()	Первую букву каждого слова переводит в верхний регистр, а все остальные в нижний
format()	Форматирование строки
upper()	Преобразование строки к верхнему регистру
lower()	Преобразование строки к нижнему регистру

Основные методы строк: capitalize

Исходная строка:

```
'GCTGGGGGGCTATAAAAAGAGGAGGCGCAGCGGAGAGACAGAGACGGACAAAGACCAAGAGAGCGAAACCAGAGAGGAGCAAGAAGCAGCAA
ACACCAGACCATTCTTTGACCGACTCCAGCATGGGCTCCTCCGCCATC'
```

После использования метода capitalize():

```
'Gctgggggggctataaaaagaggaggcgcagcggagagacagagacggacaaagaccaagagagcgaaaccagagaggagcaagaagcagcaa
acaccagaccattctttgaccgactccagcatgggctcctccgccatc'
```

Основные методы строк: swapcase

Исходная строка:

```
'GCTGGGGGGCTATAAAAAGAGGAGGCGCAGCGGAGAGACAGAGACGGACAAAGACCAAGAGAGCGAAACCAGAGAGGAGCAAGAAGCAGCAA
ACACCAGACCATTCTTTGACCGACTCCAGCATGGGCTCCTCCGCCATC'
```

После использования метода swapcase():

```
'gctgggggggctataaaaagaggaggcgcagcggagagacagagacggacaaagaccaagagagcgaaaccagagaggagcaagaagcagcaa
acaccagaccattctttgaccgactccagcatgggctcctccgccatc'
```

Форматирование строк

Метод	Что делает
<	Выравнивание объекта по левому краю
>	Выравнивание объекта по правому краю
=	Заполнитель будет после знака, но перед цифрами. Работает только с числовыми типами
^	Выравнивание по центру

Форматирование строк: метод format

In [112]:
```
# Используем самый простой вариат использования format - вставить слово в нужную позицию
string_for_formating = 'Данный курс проводится в городе {}'.format('Сочи')
string_for_formating
```

Out [112]:
```
'Данный курс проводится в городе Сочи'
```

Также format позволяет встраивать любое количество объектов в строку. В этом случае нужно просто указывать встраиваемый объект, который передается format. Гораздо проще это понять на примере

In [113]:
```
string_for_formating1 = 'Данный {0} проводится в {1} {2}'.format('курс', 'городе', 'Сочи')
string_for_formating1
```

Out [113]:
```
'Данный курс проводится в городе Сочи'
```

Форматирование строк: метод format

In [115]:

```
# Использования переменной для передачи в format
city = 'Сочи'
string_for_formating3 = 'Данный курс проводится в городе {}'.format(city)
string_for_formating3
```

Out [115]:

```
'Данный курс проводится в городе Сочи'
```

In [116]:

```
# Также переменную можно создать и в самом format, причем можно создавать их сколько угодно. Главное помнить про длину строки
string_for_formating4 = 'Данный {course} проводится в городе {city}'.format(course='курс', city='Сочи')
string_for_formating4
```

Out [116]:

```
'Данный курс проводится в городе Сочи'
```

Форматирование строк: метод format

In [117]:

```
# Если в коде необходимо сделать и format и вставить фигурные скобочки, чтобы они именно отображались в строке, для этого
#можно использовать экранирование, просто поместив немобходимый эелментв в еще одни фигурные скобочки как в примере

string_for_formating5 = 'Данный {{course}} проводится в городе {city}'.format(course='курс', city='Сочи')
string_for_formating5
```

Out [117]:

```
'Данный {course} проводится в городе Сочи'
```

Форматирование строк: метод format

In [15]:
```
#Примеры форматирования: положения course по центру с 10 пробельными симолами по краям
string_for_formating6 = 'Данный {course:^10} проводится в городе {city}'.format(course='курс', city='Сочи')
string_for_formating6
```

Out [15]:
```
'Данный    курс    проводится в городе Сочи'
```

In [120]:
```
#Примеры форматирования: положения course выравнивание по левому краю с 10 нижними подчеркиваниями вместо пробела
string_for_formating7 = 'Данный {course:_<10} проводится в городе {city}'.format(course='курс', city='Сочи')
string_for_formating7
```

Out [120]:
```
'Данный курс______ проводится в городе Сочи'
```

Форматирование строк: f-строки

In [5]:

```
# Форматирование с помощью f
course = 'курс по программированию на Python'
city = 'Сочи'

string_for_formating8 = f'Данный {course} проводится в городе {city}'
string_for_formating8
```

Out [5]:

```
'Данный курс по программированию на Python проводится в городе Сочи'
```