Строки (str)

Строка – это неизменяемая последовательность юникодных символов

In [23]:
```
example_string = "Курс про Python на Coursera"
print(example_string)
```

Курс про Python на Coursera

In [22]:
```
print(type(example_string))
```

<class 'str'>

In [20]:
```
example_string = 'Курс про "Python" на "Coursera"'
print(example_string)
```

Курс про "Python" на "Coursera"

In [23]:
```
example_string = "Курс про \"Python\" на \"Coursera\""
print(example_string)
```

Курс про "Python" на "Coursera"

"Сырые" (r-строки):

In [27]:
```
example_string = "Файл на диске c:\\\\"
print(example_string)

example_string = r"Файл на диске c:\\"
print(example_string)
```

Файл на диске c:\\
Файл на диске c:\\

Как разбить объявление длинной строки:

In [4]:
```
example_string = "Perl – это тот язык, который одинаково " \
                 "выглядит как до, так и после RSA шифрования." \
                 "(Keith Bostic)"
print(example_string)
```

Perl – это тот язык, который одинаково выглядит как до, так и после RSA шифрования.(Keith Bostic)

In [23]:
```
example_string = """
Есть всего два типа языков программирования: те, на которые
люди всё время ругаются, и те, которые никто не использует.

Bjarne Stroustrup
"""
print(example_string)
```

Есть всего два типа языков программирования:
те, на которые люди всё время ругаются, и те,
которые никто не используют.

Bjarne Stroustrup

Как объединить 2 строки в одну?

In [14]:
```
"Можно" + " просто " + "складывать" + " строки"
```

Out[14]: "Можно просто складывать строки"

In [15]:
```
"Даже умножать!" * 3
```

Out[15]: 'Даже умножать!Даже умножать!Даже умножать!'

Строки неизменяемые!

In [15]:
```
example_string = "Привет"
print(id(example_string))

example_string += ", Мир!"
print(id(example_string))
```

4379064296
4379064192

Срезы строк [start:stop:step]

In [63]:
```
example_string = "Курс про Python на Coursera"
example_string[9:]
```

Out[63]: 'Python на Coursera'

In [25]:
```
example_string = "Курс про Python на Coursera"
example_string[9:15]
```

Out[25]: 'Python'

In [26]:
```
example_string = "Курс про Python на Coursera"
example_string[-8:]
```

Out[26]: 'Coursera'

Использование step:

In [12]:
```
example_string = "0123456789"
example_string[::2]
```

Out[12]: '02468'

In [13]:
```
example_string = "Москва"
example_string[::-1]
```

Out[13]: 'авскоМ'

У строк есть методы:

In [28]:
```
quote = """Болтовня ничего не стоит. Покажите мне код.

Linus Torvalds
"""
quote.count("o")
```

Out[28]: 6

In [25]:
```
"москва".capitalize()
```

Out[25]: 'Москва'

In [29]:
```
"2017".isdigit()
```

Out[29]: True

Оператор in позволяет проверить наличие подстроки в строке:

In [30]:
```
"3.14" in "число Пи = 3.1415926"
```

Out[30]: True

In [18]:
```
"Алексей" in "Александр Пушкин"
```

Out[18]: False

Выражение **for .. in** позволяет итерироваться по строке:

In [23]:
```
example_string = "Привет"
for letter in example_string:
    print("Буква", letter)
```

Буква П
Буква р
Буква и
Буква в
Буква е
Буква т

Конвертация типов:

In [18]:
```
num = 999.01

num_string = str(num)

print(type(num_string))
num_string
```

<class 'str'>

Out[18]: '999.01'

In [19]:
```
bool("Непустая строка")
```

Out[19]: True

In [20]:
```
bool("")
```

Out[20]: False

In [22]:
```
name = ""

if not name:
    print("Имя не заполнено!")
```

Имя не заполнено!

Форматирование строк

1-ый способ форматирования:

In [155]:
```
template = "%s – главное достоинство программиста. (%s)"
template % ("Энтер", "Larry Wall")
```

Out[155]: 'Энтер – главное достоинство программиста. (Larry Wall)'

https://docs.python.org/3/library/string.html#format-specification-mini-language

2-ой способ:

In [56]:
```
"{} не лгут, но {} пользуется формулами. ({})".format(
    "Цифры", "люди", "Robert A. Heinlein"
)
```

Out[56]: 'Цифры не лгут, но люди пользуется формулами. (Robert A. Heinlein)'

Еще способ:

In [57]:
```
"{num} Кб должно хватить для любых задач. ({author})".format(
    num=640, author="Bill Gates"
)
```

Out[57]: '640 Кб должно хватить для любых задач. (Bill Gates)'

И еще f-стринги, Python >= 3.6:

In [58]:
```
subject = "оптимизация"
author = "Donald Knuth"

f"Преждевременная {subject} – корень всех зол. ({author})"
```

Out[58]: 'Преждевременная оптимизация – корень всех зол. (Donald Knuth)'

Модификаторы форматирования:

In [59]:
```
num = 8
f"Binary: {num:4b}"
```

Out[59]: 'Binary: 0b1000'

In [4]:
```
num = 2 / 3
print(num)

print(f"{num:.3f}")
```

0.6666666666666666
0.667

Больше описания и примеров в документации:

https://docs.python.org/3/library/string.html

Встроенная функция input()

Позволяет получить ввод пользователя в виде строки

In [60]:
```
name = input("Введи свое имя: ")

f"Привет, {name}!"
```

Введи свое имя: Александр

Out[60]: 'Привет, Александр!'

Байтовые строки (bytes)

Байт - минимальная единица хранения и обработки цифровой информации. Последовательность байт представляет собой какую-либо информацию (текст, картинку, мелодию...)

Байтовая строка – это неизменяемая последовательность чисел от 0 до 255.

b-литерал для объявления байтовой строки:

In [17]:
```
example_bytes = b"Hello"
print(type(example_bytes))
```

<class 'bytes'>

In [47]:
```
for element in example_bytes:
    print(element)
```

104
101
108
108
111

In [48]:
```
example_bytes = b"привет"
```

```
  File "<ipython-input-48-f18cf5690d99>", line 1
    example_bytes = b"привет"
                    ^
SyntaxError: bytes can only contain ASCII literal characters.
```

Bytes literals are always prefixed with 'b' or 'B'; they produce an instance of the bytes type instead of the str type. They may only contain ASCII characters; bytes with a numeric value of 128 or greater must be expressed with escapes.

In [43]:
```
example_string = "привет"
print(type(example_string))
print(example_string)
```

<class 'str'>
привет

In [42]:
```
encoded_string = example_string.encode(encoding="utf-8")
print(encoded_string)
print(type(encoded_string))
```

b'\xd0\xbf\xd1\x80\xd0\xb8\xd0\xb2\xd0\xb5\xd1\x82'
<class 'bytes'>

Буква	Кодировка	hex	dec (bytes)	dec	binary
п	UTF-8	D0 BF	208 191	53439	11010000 10111111

буква п - https://unicode-table.com/ru/043F/

Декодируем байты обратно в строку:

In [45]:
```
decoded_string = encoded_string.decode()
print(decoded_string)
```

привет

В этом видео:

- Поговорили о строковых типах в Python
- Рассмотрели способы форматирования строк
- Узнали как получить ввод пользователя в виде строки
- Посмотрели как работать с последовательностями - срезы, итерация

In [ ]:
```

```