9.6 Форматирование строк

Шаг 1

Тема урока: форматирование строк

1. Строковый метод `format()`
2. f-строки

Аннотация. Строковый тип данных, основные методы форматирования строк.

Метод format()

Хранить строки в переменных удобно, но часто бывает необходимо собирать строки из других объектов (строк, чисел и т.д.) и выполнять с ними нужные манипуляции. Для этой цели можно воспользоваться механизмом **форматирования строк**.

Рассмотрим следующий код:

```
birth_year = 1992
text = 'My name is Timur, I was born in ' + birth_year

print(text)
```

Такой код приводит к ошибке во время выполнения программы, поскольку мы пытаемся сложить число и строку:

```
TypeError: can only concatenate str (not "int") to str
```

Для решения такой проблемы мы можем использовать функцию `str()`, которая преобразует числовое значение в строку.

Приведенный ниже код:

```
birth_year = 1992
text = 'My name is Timur, I was born in ' + str(birth_year)

print(text)
```

выводит:

```
My name is Timur, I was born in 1992
```

Такой код работает, но каждый раз преобразовывать число в строку не очень удобно. Для более наглядного форматирования мы можем использовать строковый метод `format()`.

Предыдущий код можно переписать в виде:

```
birth_year = 1992
text = 'My name is Timur, I was born in {}'.format(birth_year)

print(text)
```

Мы передаем необходимые параметры методу `format()`, а Python ставит их вместо фигурных скобок `{}` – **заполнителей**. Мы можем создавать сколько угодно заполнителей.

Приведенный ниже код:

```
birth_year = 1992
name = 'Timur'
profession = 'math teacher'
text = 'My name is {}, I was born in {}, I work as a {}.'.format(name, birth_year, profession)

print(text)
```

выводит:

```
My name is Timur, I was born in 1992, I work as a math teacher.
```

Для наглядности и гибкости форматирования мы можем использовать порядковый номер в заполнителе: `{0}`, `{1}`, `{2}` и т.д. Такой номер определяет позицию параметра, переданного методу `format()` (нумерация начинается с нуля):

```
birth_year = 1992
name = 'Timur'
profession = 'math teacher'
text = 'My name is {0}, I was born in {1}, I work as a {2}.'.format(name, birth_year, profession)

print(text)
```

Параметр `name` встает в заполнителе `{0}`, параметр `birth_year` – в заполнителе `{1}` и т.д. Мы можем использовать одно и то же число в нескольких заполнителях или не использовать совсем, а также мы можем использовать числа в разном порядке.

Приведенный ниже код:

```
name = 'Timur'
city = 'Moscow'
text1 = 'My name is {0}-{0}-{0}!'.format(name, city)
text2 = '{1} is my city and {0} is my name!'.format(name, city)

print(text1)
print(text2)
```

выводит:

```
My name is Timur-Timur-Timur!
Moscow is my city and Timur is my name!
```

f-строки

Метод `format()` хорошо справляется с задачей форматирования строк, однако если параметров много, то код может показаться немного избыточным.

Приведенный ниже код:

```
first_name = 'Taylor'
second_name = 'Swift'
country = 'USA'
birth_date = '1989/12/13'
birth_place = 'West Reading, Pennsylvania'
text = '{0} {1} is a very famous singer from the {2}. She was born on {3} in {4}.'.format(first_name,
second_name, country, birth_date, birth_place)

print(text)
```

выводит:

```
Taylor Swift is a very famous singer from the USA. She was born on 1989/12/13 in West Reading,
Pennsylvania.
```

В Python 3.6 появилась новая разновидность строк — f-строки. Если поставить перед строкой префикс `f`, в заполнители можно будет включить код, например, имя переменной или любые другие выражения. f-строки обеспечивают чистый и интуитивно понятный способ форматирования строк.

Предыдущий код можно записать в виде:

```
first_name = 'Taylor'
last_name = 'Swift'
country = 'USA'
birth_date = '1989/12/13'
birth_place = 'West Reading, Pennsylvania'
text = f'{first_name} {last_name} is a very famous singer from the {country}. She was born on {birth_date}
in {birth_place}.'

print(text)
```

На место заполнителя `{first_name}` встает значение переменной `first_name`, на место заполнителя `{last_name}` встает значение переменной `last_name` и т.д.

Как уже говорилось, помимо переменных в заполнителях f-строк мы можем использовать выражения.

Приведенный ниже код:

```
print(f'5 + 2 = {5 + 2}')
print(f'5 - 2 = {5 - 2}')
print(f'5 * 2 = {5 * 2}')
print(f'5 / 2 = {5 / 2}')
```

выводит:

```
5 + 2 = 7
5 - 2 = 3
5 * 2 = 10
5 / 2 = 2.5
```

Примечания

Примечание 1. Почитать подробнее про форматирование строк можно в официальной документации по ссылке (https://docs.python.org/3/library/string.html#formatstrings). Отдельно можно почитать на английском языке про метод `format()` и f-строки по ссылке (https://docs.python.org/3/library/stdtypes.html#str.format) и по ссылке (https://docs.python.org/3/reference/lexical_analysis.html#f-strings).

Примечание 2. На русском языке можно почитать про форматирование строк по ссылке (https://docs-python.ru/tutorial/operatsii-tekstovymi-strokami-str-python/metod-str-format/) и по ссылке (https://docs-python.ru/tutorial/operatsii-tekstovymi-strokami-str-python/stroki-formatirovannye-stroki/).

Примечание 3. Про эволюцию форматирования в Python вы можете почитать в нашем официальном блоге на Хабре по ссылке (https://habr.com/ru/articles/828396/).

Примечание 4. До версии Python 3.12 повторное использование тех же кавычек, что и окружающие f-строку, вызывает ошибку синтаксиса.

Приведенный ниже код:

```
text = f'{'London'} is the capital of {'England'} and the {'United Kingdom'}'
```

приводит к возникновению ошибки:

```
SyntaxError: f-string: expecting '}'
```

В версии Python 3.12 это ограничение сняли. Обратите внимание, что на данный момент на Stepik **недоступна версия** Python 3.12, и если вы копируете код из своей IDE (в которой, возможно, уже стоит Python версии 3.12 и выше), пожалуйста, проверяйте, что вы не используете повторно одни кавычки.

❤️ Happy Pythoning!

Шаг 2

Что покажет приведённый ниже код?

```
planet = 'Arrakis'
bad_guys = 'Harkonnens'
text = 'The desert planet {}, rich in valuable spice, is exploited by cruel {}.'.format(planet, bad_guys)

print(text)
```

Чтобы решить это задание откройте https://stepik.org/lesson/1209103/step/2

Шаг 3

Что покажет приведённый ниже код?

```
name = 'Leto Atreides'
planet = 'Caladan'
text = 'Duke {1} is the ruler of the planet {0}.'.format(planet, name)

print(text)
```

Чтобы решить это задание откройте https://stepik.org/lesson/1209103/step/3

Шаг 4

Что покажет приведённый ниже код?

```
name = 'Dune'
text = f'The novel "{name}" was published in 1965 by Frank Herbert.'

print(text)
```

Чтобы решить это задание откройте https://stepik.org/lesson/1209103/step/4

Шаг 5

Что покажет приведённый ниже код?

```
name = 'Imperium'
text = 'For the {name} spice is used by the navigators to find safe paths between the stars.'

print(text)
```

Чтобы решить это задание откройте https://stepik.org/lesson/1209103/step/5

Шаг 6

Что покажет приведённый ниже код (в версиях Python **до** 3.12)?

```
text = f'By Imperial decree, Leto Atreides is now the fief of the planet {'Arrakis'}.'
print(text)
```

Чтобы решить это задание откройте https://stepik.org/lesson/1209103/step/6

Шаг 7

Используя метод `format()`, дополните приведённый ниже код так, чтобы он вывел текст:

```
In 2010, someone paid 10k Bitcoin for two pizzas.
```

Sample Input:

Sample Output:

```
In 2010, someone paid 10k Bitcoin for two pizzas.
```

Чтобы решить это задание откройте https://stepik.org/lesson/1209103/step/7

Шаг 8

Используя f-строку, дополните приведённый ниже код так, чтобы он вывел текст:

```
In 2010, someone paid 10K Bitcoin for two pizzas.
```

Sample Input:

Sample Output:

```
In 2010, someone paid 10K Bitcoin for two pizzas.
```

Чтобы решить это задание откройте https://stepik.org/lesson/1209103/step/8

Шаг 9

Курсы валют 💹

Вследствие кибератаки на банк «Разбогатеем вместе» сломался алгоритм, выводящий курсы валют для определённой даты в мобильном приложении. Технический отдел банка просит вас исправить ситуацию и наладить вывод. На вход программе подаются следующие значения:

- дата (в формате ДД-ММ-ГГГГ)
- курс евро (сколько российских рублей стоит 1 евро)
- курс юаня (сколько российских рублей стоит 1 юань)

Напишите программу, которая выводит строку, показывающую, сколько российских рублей стоит 1 евро и 1 юань на указанную дату в формате:

```
На <дата>: 1€ = <курс евро>₽, 1¥ = <курс юаня>₽
```

Формат входных данных
На вход программе подаются три строки (каждая на отдельной строке): дата, курс евро и курс юаня.

Формат выходных данных
Программа должна вывести текст в соответствии с условием задачи.

▶ Тестовые данные 🟢

Sample Input 1:

```
16-03-2024
99.9718
12.7328
```

Sample Output 1:

```
На 16-03-2024: 1€ = 99.9718₽, 1¥ = 12.7328₽
```

Sample Input 2:

```
12-05-2016
75.4505
10.1721
```

Sample Output 2:

```
На 12-05-2016: 1€ = 75.4505₽, 1¥ = 10.1721₽
```

Чтобы решить это задание откройте https://stepik.org/lesson/1209103/step/9

Шаг 10

Сумма кубов VS Куб суммы

Очень часто студенты "Поколения Python" путают понятия «сумма кубов» и «куб суммы». Для того чтобы внести ясность в этот извечный математический вопрос, предлагаем вам решить следующую задачу.

На вход программе подаются два целых числа a и b. Ваша программа должна посчитать для этих чисел сумму их кубов и куб их суммы и вывести результат вычислений в следующем формате:

```
Для чисел <число a> и <число b>:
  Сумма кубов: <число a>**3 + <число b>**3 = <сумма кубов a и b>
  Куб суммы: (<число a> + <число b>)**3 = <куб суммы a и b>
```

Формат входных данных

На вход программе подаются два целых числа a и b, каждое на отдельной строке.

Формат выходных данных

Программа должна вывести текст в соответствии с условием задачи.

Примечание. Не перепутайте сумму кубов и куб суммы. 😉

▶ Тестовые данные 🟢

Sample Input 1:

```
1
2
```

Sample Output 1:

```
Для чисел 1 и 2:
  Сумма кубов: 1**3 + 2**3 = 9
  Куб суммы: (1 + 2)**3 = 27
```

Sample Input 2:

```
-2
0
```

Sample Output 2:

```
Для чисел -2 и 0:
  Сумма кубов: -2**3 + 0**3 = -8
  Куб суммы: (-2 + 0)**3 = -8
```

Чтобы решить это задание откройте https://stepik.org/lesson/1209103/step/10

Шаг 11

(Не) Активное похудение 🏃 🌶️

Гвидо, засевший за компьютером и не ведущий активный образ жизни, «немного» поднабрал в весе. Осталось всего 60 дней до лета, а хочется быть в форме. Вот Гвидо и решился на похудение. Все дни до лета он пронумеровал от 1 до 60 (включительно). Перед началом похудения у Гвидо был вес 100 кг, а своей целью он поставил достичь веса 88 кг (или меньше). Он решил худеть на одну и ту же массу ежедневно.

Напишите программу, которая принимает на вход текущий день и текущий вес Гвидо. Программа должна вывести фразу:

- «Все идет по плану» (без кавычек), если Гвидо удаётся держать планку в похудении и его вес ниже либо равен тому, который он запланировал на текущий день
- «Что-то пошло не так» (без кавычек), если Гвидо не очень старается и его вес выше того, который он запланировал на текущий день

Также программа должна вывести информацию о номере дня похудения, текущем весе Гвидо и цели по весу на текущий день в формате:

```
#<номер дня> ДЕНЬ: ТЕКУЩИЙ ВЕС = <текущий вес Гвидо> кг, ЦЕЛЬ по ВЕСУ = <цель по весу на текущий день> кг
```

Формат входных данных

На вход программе подаются два числа (каждое на отдельной строке): номер дня похудения (целое число) и текущий вес Гвидо (действительное число).

Формат выходных данных

Программа должна вывести текст в соответствии с условием задачи.

Примечание. В 1-й день похудения Гвидо уже должен похудеть (см. 1 тест).

▶ Подсказка

▶ Тестовые данные

Sample Input 1:

```
1
99.9
```

Sample Output 1:

```
Что-то пошло не так
#1 ДЕНЬ: ТЕКУЩИЙ ВЕС = 99.9 кг, ЦЕЛЬ по ВЕСУ = 99.8 кг
```

Sample Input 2:

```
1
99.16
```

Sample Output 2:

```
Все идет по плану
#1 ДЕНЬ: ТЕКУЩИЙ ВЕС = 99.16 кг, ЦЕЛЬ по ВЕСУ = 99.8 кг
```