IT CAREER HUB

Урок 12.
Строки: форматирование

Форматирование строк 2
Задания для закрепления 1 5
Метод format 6
Задания для закрепления 2 10
Форматирование чисел 11
Задания для закрепления 3 17
Ответы на задания 19
Практические задания 20

IT CAREER HUB

Форматирование строк

Форматирование строк — это способ вставки значений переменных или результатов выражений внутрь строки.

В Python существует несколько способов форматирования строк: с использованием оператора `%`, метода `format()`, а также f-строк (интерполяции строк).

C-style форматирование

`C-style` форматирование строк в Python использует оператор `%` для вставки значений в строку. Этот способ форматирования напоминает стиль форматирования строк в языке программирования C и был одним из первых способов форматирования в Python.

Синтаксис:

```python
"форматирующая строка" % (значения)
```

- форматирующая строка содержит спецификаторы, начинающиеся с символа `%`, которые указывают, как следует форматировать соответствующие значения.
- значения — это переменные или выражения, которые будут подставлены в строку в том же порядке.

Основные спецификаторы:

Спецификатор	Описание
`%s`	Строка
`%d`	Целое число
`%f`	Число с плавающей точкой
`%.2f`	Число с плавающей точкой, округлённое до 2 знаков после запятой

Примеры использования

1. Форматирование строки и целого числа:

```python
name = "Alice"

age = 30

text = "My name is %s and I am %d years old." % (name, age)

print(text)
```

2. Форматирование числа с плавающей точкой:

```python
pi = 3.14159

text = "The value of pi is approximately %.2f." % pi

print(text)
```

Преобразование типов в C-style

В C-style форматировании можно легко преобразовывать числа в строки с помощью спецификаторов формата. Например, использование `%s` автоматически преобразует любое число в строку. Но при этом невозможно преобразовать строку в число, это вызовет ошибку `TypeError`.

Пример

```python
num = 42

text = "The number is %s." % num
```

```
print(text)


text = "42"  # Число в формате str


formatted = "This will cause an error: %d" % text

print(formatted)
```

IT CAREER HUB

Задания для закрепления 1

1. Какой результат будет выведен при выполнении следующего кода?

```python
name = "Alice"

age = 30

text = "My name is %d and I am %s years old." % (name, age)

print(text)
```

a. "My name is Alice and I am 30 years old."
b. "My name is %s and I am %d years old."
c. Ошибка
d. "My name is Alice and I am %d years old."

Посмотреть ответ

2. Какой результат будет выведен при выполнении следующего кода?

```python
pi = 3.14159

text = "The value of pi is approximately %.2f." % pi

print(text)
```

a. "The value of pi is approximately 3.14."
b. "The value of pi is approximately 3.14159."
c. Ошибка
d. "The value of pi is approximately %.2f."

Посмотреть ответ

IT CAREER HUB

Метод format

Метод format() — это гибкий способ форматирования строк, который позволяет вставлять значения в строку с использованием фигурных скобок {} в качестве плейсхолдеров (места для вставки значений).

Этот метод является более современным и предпочтительным по сравнению с C-style форматированием через %.

Синтаксис:

```python
"строка с {} внутри".format(значение1, значение2, ...)
```

- Внутри строки используются фигурные скобки `{}`, которые будут заменены переданными значениями из метода `format()`.
- Можно использовать позиционные аргументы, именованные аргументы, а также указание форматов для чисел и строк.

Основные способы использования:

1. **Позиционные аргументы** — значения подставляются в порядке их передачи.

```python
name = "Alice"

age = 30

text = "My name is {} and I am {} years old."

print(text.format(name, age))
```

2. **Именованные аргументы** — каждому плейсхолдеру можно присвоить имя, что делает форматирование более понятным. При этом можно несколько раз использовать то же имя.

```python
text = "My name is {name} and I am {age} years old. Are you also {age} years old?"

print(text.format(name="Bob", age=25))
```

3. **Использование индексов** — можно явно указать, какой аргумент вставить в каждое место. Индексы также можно использовать несколько раз.

```python
text = "Her name is {0} and she is {1} years old. {0} loves Python."

print(text.format("Anna", 28))
```

4. **Комбинирование позиционных и именованных аргументов** — можно использовать как позиционные, так и именованные аргументы одновременно, что даёт большую гибкость в форматировании строк.

```python
text = "The {0} is {color}."

print(text.format("sky", color="blue"))
```

f-строки

f-строки — это удобный способ форматирования строк, который был введён в Python 3.6.

Они позволяют вставлять переменные и выражения в строку напрямую в строку, используя фигурные скобки {}. f-строки делают код более лаконичным и удобным по сравнению с другими методами форматирования.

Синтаксис:

Чтобы использовать f-строки, перед строкой нужно добавить префикс f, а переменные или выражения помещаются напрямую в фигурные скобки {} внутри строки.

```python
f"текст {переменная} текст {выражение}"
```

Примеры использования

1. Вставка переменных:

```python
name = "Alice"

age = 25

text = f"My name is {name} and I am {age} years old."

print(text)
```

2. Вставка выражений:

```python
x = 10

y = 20

text = f"The sum of {x} and {y} is {x + y}."

print(text)
```

3. Вставка вызова функций и методов:

```python
text = "Python"

text_info = f"The length of '{text}' is {len(text)} and its uppercase version is {text.upper()}."

print(text_info)
```

4. Вставка в многострочную строку:

```python
name = "Charlie"

age = 30

text = f"""Info

Name: {name}

Age: {age}

"""

print(text)
```

Преимущества f-строк:

- **Удобство:** Позволяют вставлять переменные и выражения прямо в строку.
- **Читаемость:** Делают код более понятным и лаконичным.
- **Гибкость:** Поддерживают любые выражения внутри `{}`.
- **Быстрота:** Работают быстрее, чем другие способы форматирования строк, такие как `%` и `format()`.

IT CAREER HUB

Задания для закрепления 2

1. Какой результат будет выведен при выполнении следующего кода?

```python
text = f"The sum of {10} and {20} is {10 + 20}."

print(text)
```

a. "The sum of 10 and 20 is 30."
b. "The sum of 10 and 20 is {10 + 20}."
c. Ошибка
d. "The sum of {10} and {20} is {10 + 20}."

Посмотреть ответ

2. Какой результат будет выведен при выполнении следующего кода?

```python
text = "The {0} is {color}.".format("sky", color="blue")

print(text)
```

a. "The sky is color."
b. "The {0} is {color}."
c. "The sky is blue."
d. "The 0 is color."

Посмотреть ответ

IT CAREER HUB

Форматирование чисел

Метод `format()` и f-строки предоставляют гибкие возможности для форматирования чисел. Эти два метода работают схожим образом, но f-строки более удобны и лаконичны.

Основные способы форматирования чисел:

1. Ограничение количества знаков после запятой: Используйте спецификатор `:.nf`, где `n` — это количество знаков после запятой, `f` — спецификатор для `float`.

Примеры

```python
pi = 3.14159

# f-строки

text_fstring = f"Pi rounded to 2 decimal places is {pi:.2f}"

# Метод format()

text_format1 = "Pi rounded to 2 decimal places is {:.2f}".format(pi)

text_format2 = "Pi rounded to 2 decimal places is {0:.2f}".format(pi)

text_format3 = "Pi rounded to 2 decimal places is {num:.2f}".format(num=pi)

print(text_fstring)

print(text_format1)

print(text_format2)

print(text_format3)
```

2. Форматирование с разделителями тысяч: Используйте спецификатор `:,` чтобы добавить разделители тысяч в больших числах.

Примеры

```python
large_number = 1234567890

# f-строки

text_fstring = f"The number with thousand separators: {large_number:,}"

# Метод format()

text_format = "The number with thousand separators: {:,}".format(large_number)

print(text_fstring)

print(text_format)
```

Выравнивание и ширина поля

И в методе `format()`, и в f-строках можно задавать выравнивание текста и чисел, а также ширину поля для их отображения. Это полезно при работе с форматированным выводом, например, в таблицах или структурированных данных.

Основные спецификаторы для выравнивания и ширины поля:

- `>` — выравнивание по правому краю.
- `<` — выравнивание по левому краю.
- `^` — выравнивание по центру.
- Число после символом выравнивания задаёт минимальную ширину поля, которая будет выделена для значения.

Примеры использования

1. Выравнивание по правому краю:
    - Чтобы выравнять значение по правому краю, используйте символ `>`.

Пример

```python
# f-строки

text_fstring = f"start_{'text':>10}_end"

# Метод format()

text_format = "start_{:>10}_end"

print(text_fstring)

print(text_format.format("text"))
```

2. **Выравнивание по левому краю:**
    - Для выравнивания по левому краю используйте символ `<`.

Пример

```python
# f-строки

text_fstring = f"start_{'text':<10}_end"

# Метод format()

text_format = "start_{:<10}_end"

print(text_fstring)

print(text_format.format("text"))
```

3. **Выравнивание по центру:**
    - Для выравнивания по центру используйте символ `^`.

Пример

```python
# f-строки

text_fstring = f"start_{'text':^10}_end"

# Метод format()

text_format = "start_{:^10}_end"

print(text_fstring)

print(text_format.format("text"))
```

4. Задание минимальной ширины поля для чисел:
    - Можно задать только минимальную ширину для чисел, добавив число после символа двоеточия.
    - Для чисел выравнивание по умолчанию будет по правому краю.
    - Для строк выравнивание по умолчанию будет по левому краю.

Пример

```python
number = 40
text = 'hi'
# f-строки
text_fstring = f"start_{number:5}_end"
# Метод format()
text_format = "start_{:5}_end"
print(text_fstring)
print(text_format.format(text))
```

5. Выравнивание чисел с заполнением другими символами:
    - Python позволяет выравнивать строки и числа не только с помощью пробелов, но и с заполнением другими символами, например, нулями или любыми другими символами, которые вы укажете.
    - Для этого в f-строках можно указать символ заполнения перед символом выравнивания (`<`, `>`, `^`) или до значения.

Пример

```python
# Заполнение нулями

number = 40

text = f"{number:0>5}"

print(text)



# Заполнение нижним подчеркиванием

text = f"{'Python':_^10}"

print(text)
```

Методы выравнивания строк

В Python для выравнивания строк существуют три метода (`center`, `ljust`, `rjust`), которые помогают располагать строки по центру, слева или справа с использованием заданной ширины и, при необходимости, с заполнением оставшегося пространства символами.

1. `str.ljust(width[, fillchar])` — выравнивание строки по левому краю.
2. `str.rjust(width[, fillchar])` — выравнивание строки по правому краю.
3. `str.center(width[, fillchar])` — выравнивание строки по центру.

Синтаксис:

```python
str.rjust(width[, fillchar])
```

- **width** — минимальная ширина строки.
- **fillchar (опционально)** — символ для заполнения свободного пространства (по умолчанию — пробел).

Примеры использования

```python
text = "Python"


# ljust(): выравнивание по левому краю

print(text.ljust(15))

print(text.ljust(15, '-'))


# rjust(): выравнивание по правому краю

print(text.rjust(15))

print(text.rjust(15, '-'))


# center(): выравнивание по центру

print(text.center(15))

print(text.center(15, '-'))
```

IT CAREER HUB

Задания для закрепления 3

1. Какой результат будет выведен при выполнении следующего кода?

```python
number = 1234.5678

print(f"Formatted number: {number:.2f}")
```

a. "Formatted number: 1234.5678"
b. "Formatted number: 1234.57"
c. "Formatted number: 1234.56"
d. "Formatted number: 1234.56f"

Посмотреть ответ

2. Какой результат будет выведен при выполнении следующего кода?

```python
large_number = 9876543210

print(f"The number is: {large_number:,}")
```

a. "The number is: 9876,543,210"
b. "The number is: 9,876543210"
c. "The number is: 9,876,543,210"
d. "The number is: 9876543210,"

Посмотреть ответ

3. Какой результат будет выведен при выполнении следующего кода?

```python
text = f"{'Python':_^10}"
```

```
print(text)
```

a. "Python____"
b. "__Python__"
c. "Python "
d. Ошибка

Посмотреть ответ

IT CAREER HUB

Ответы на задания

Задания на закрепление 1	Вернуться к заданиям
1. Результат выполнения кода	Ответ: c
2. Результат выполнения кода	Ответ: a
Задания на закрепление 2	Вернуться к заданиям
1. Результат выполнения кода	Ответ: a
2. Результат выполнения кода	Ответ: c
Задания на закрепление 3	Вернуться к заданиям
1. Результат выполнения кода	Ответ: b
2. Результат выполнения кода	Ответ: c
3. Результат выполнения кода	Ответ: b

IT CAREER HUB

Практическая работа

1. Счетчик слов

Напишите программу, которая обрабатывает строку и выводит её, добавив к каждому слову его порядковый номер, выравнивая текст по левому краю с длиной в 15 символов. Слова выводите с большой буквы.

Пример вывода:

```python
Введите строку: Hello world Python is great
1. Hello
2. World
3. Python
4. Is
5. Great
```

Решение:

```python
text = "Hello world Python is great"
words = text.split()

i = 1
for word in words:
    print(f"{i}. {word.capitalize():<15}")
    i += 1
```

IT CAREER HUB

2. Формат даты

Напишите программу, которая принимает дату в виде числа, месяца и года, а затем выводит её в формате "dd/mm/yyyy", где день и месяц всегда состоят из двух цифра.

Пример вывода:

```
Python
Введите день: 3
Введите месяц: 7
Введите год: 2024
Дата: 03/07/2024
```

Решение:

```python
day = int(input("Введите день: "))
month = int(input("Введите месяц: "))
year = int(input("Введите год: "))

formatted_date = "{:02}/{:02}/{}".format(day, month, year)
print(f"Дата: {formatted_date}")
```