Конспект: Числа и строки в Python. Конструкция IF.

В.И. Фирсанова

24 сентября 2024

1 Числа

1.1 Комплексные числа: `complex`

Комплексные числа представляются в виде $a + bi$, где a — вещественная часть, а bi — мнимая часть, которая состоит из вещественного числа b и мнимой единицы i, равной $\sqrt{-1}$):

```
x = 3+5j
y = 5j

print(type(x))  # <class 'complex'>
print(type(y))  # <class 'complex'>
```

1.2 Числа с плавающей точкой: `float`

Числа с плавающей точкой соответствуют вещественным числам, однако не являются ими из-за особенностей представления чисел в памяти вычислительного устройства:

```
a = 0.1 + 0.1 + 0.1
b = 0.3
print(a == b)  # False
```

Почему $0.1+0.1+0.1$ не равно 0.3? Вещественные числа состоят из двух частей – мантиссы и порядка. Мантисса – это целое число, например, 3. Порядок – это 10 в n-ной степени. Порядок задает количество знаков после запятой. Чем больше памяти в битах выделяется на кодирование мантиссы и порядка, тем ближе наши числа к вещественным числам.

1.3 Целые числа: `int`

Целые числа не имеют дробной части. Преобразование к целочисленному виду дает округление:

```
x = 42
print(type(x))  # <class 'int'>

y = 42.1
print(int(y))  # 42
```

1.4 Булева переменная

Булев тип данных имеет два значения: `True` и `False`. В Python этот тип следует из логических выражений, например *and*, *or*, *not*:

```
3 == 3 and 3 == 3  # True
3 == 3 or 3 == 2   # True
not isinstance(3, int)  # False
4 not in [1, 2, 3]  # True

3 > 1  # True
0 == 0  # True
1 != 1 # False
2 <= 1 # False
```

2 Строки

Строка — это последовательность символов, и она является итерируемым объектом:

```
"Hello, World"
'Hello, World'
```

Строки могут содержать специальные символы, например, перенос строки:

```
'Hello, World\nHello, World'
```

Для вывода строки используем функцию `print()`:

```
print('Hello, World')  # Hello, World
```

2.1 Форматирование строк: f-strings

Использование f-строк позволяет вставлять значения переменных в строку:

```
name = "John"
print(f'Hello, {name}')  # Hello, John
```

2.2 Основные функции для работы со строками

- **Длина строки**: Функция `len()` возвращает количество символов в строке.

```
s = "Hello, World!"
print(len(s))  # 13
```

- **Приведение к верхнему и нижнему регистру**:

```
s = "Hello"
print(s.upper())  # HELLO
print(s.lower())  # hello
```

- **Капитализация строки**:

```
s = "hello"
print(s.capitalize())  # Hello
```

- **Разбиение строки**:

```
s = "one, two, three"
print(s.split(','))  # ['one', 'two', 'three']
```

- **Удаление пробелов**:

```
s = "  hello  "
print(s.strip())  # hello
```

- **Замена подстроки**:

```
s = "this is an apple"
print(s.replace("apple", "orange"))  # this is an orange
```

- **Регулярные выражения**:

```
import re
text = "Contact us at info@example.com"
cleaned_text = re.sub(r'\S+@\S+', '', text)
print(cleaned_text)  # Contact us at
```

3 Условные операторы

Условные операторы используются для выполнения кода в зависимости от истинности выражения:

```
var = 3
if var == 3:
    var += 1
print(var)  # 4
```

Оператор `else` используется для обработки случая, когда условие `if` ложно:

```
if var == 3:
    var += 1
else:
    var = 3
```

Оператор `elif` используется для проверки дополнительных условий:

```
if var == 3:
    var += 1
elif var == 4:
    var += 2
else:
    var = 3
```