Python: простые типы данных | Я.Шпора

Числовые типы

Целые числа: `int`

К типу `int` относятся все положительные и отрицательные числа без дробной части, в том числе и ноль.

Для лучшей читаемости целые числа в Python можно записывать, разделяя разряды нижним подчёркиванием: 432246123 ⇒ 432_246_123.

Числа с плавающей точкой

Число с плавающей точкой записывается следующим образом:

1. Целая часть числа.
2. Точка, отделяющая дробную часть от целой.
3. Дробная часть числа.

```
f = 1.05
```

Если у числа нет целой или дробной части, можно её не указывать:

```
# Запись, аналогичная записи b = 2.0.
b = 2.
# Запись, аналогичная записи x = 0.07.
x = .07
```

Арифметические операции с числами

Знак в Python	Название	Пример
`+`	Сложение	`1 + 2`
`-`	Вычитание	`43 - 1`
`*`	Умножение	`5 * 8`

Знак в Python	Название	Пример
`/`	Деление	`7.2 / 3`
`//`	Целочисленное деление	`40 // 3` будет равно `13`
`%`	Взятие остатка от деления	`40 % 3` будет равно `1`
`**`	Возведение в степень	`10 ** 3` будет равно `1000`

Инкрементирование — операция пошагового увеличения значения переменной на определённое число. Декрементирование — пошаговое уменьшение значения.

В Python инкрементирование можно записать через комбинированный оператор присваивания:

```
# Выражение...
steps = counter + 1
# ...можно записать через комбинированный оператор присваивания:
counter += 1
```

Комбинированные операторы присваивания применимы для всех арифметических операций: `-=`, `*=`, `/=` и других.

Приоритеты арифметических операций

- Самый приоритетный арифметический оператор — возведение в степень.
- Следующими идут операторы умножения, деления, получения остатка и целочисленное деление.
- Самые низкоприоритетные — операторы сложения и вычитания.
- При равном уровне приоритетов операторы выполняются по очереди, слева направо.
- Если рядом находится две операции возведения в степень, то последовательность их выполнения будет обратной — справа налево.
- Чтобы изменить приоритетность выполнения операций — применяют скобки. Операции в скобках выполняются в первую очередь.

Как не потерять точность в вычислениях с числами с плавающей точкой

```
input_data_1 = 3.3
input_data_2 = 4.18

# Чтобы дробная часть всех слагаемых стала равна нулю,
# каждое слагаемое нужно умножить на один и тот же множитель.
input_data = input_data_1 * 100 + input_data_2 * 100

# После получения результата нужно вернуть всё как было - разделить
# Для сокращения записи применён комбинированный оператор присваива
input_data /= 100
print(input_data)

# Вывод в терминал: 7.48
```

NoneType

Переменные с типом данных `NoneType` могут содержать лишь `None`, никаких других вариантов нет. `None` символизирует отсутствие значения как такового.

Когда используется `None`:

- Когда нужно объявить переменную, которая, по планам, точно понадобится в коде, но пока непонятно, какое значение ей присвоить.
- Когда нужно разрешить вызовы функции без явно указанного аргумента. Тогда `None` устанавливается как значение по умолчанию для параметров функции.
- Любая функция, в которой не применена инструкция `return`, возвращает `None`:

```
def rectangle_area(length, width):
    area = length * width


area = rectangle_area(5, 0)
print(area)
print(type(area))

# Вывод в терминал:
```

```
# None
# <class 'NoneType'>
```

Операторы сравнения и логический тип данных

Сравнение — это проверка выражения на истинность. Основные операции сравнения в Python:

Знак в Python	Название
`<=`	Меньше или равно
`>=`	Больше или равно
`<`	Меньше
`>`	Больше
`==`	Равно
`!=`	Не равно

При операциях сравнения возвращается значение логического типа, bool — `True` или `False`:

```
area_1 = 50
area_2 = 49

area_1 < area_2   # False
area_1 == area_2  # False
area_1 > area_2   # True
```

Чтобы проверить, равна ли переменная `None`, используется оператор `is`:

```
jar_of_vegetables = None

print(jar_of_vegetables is None) # Выведется: True
```

Когда нужно сделать серию сравнений за один раз, используется цепочка сравнений:

```
current_temperature = 19
```

```
print(10 < current_temperature < 20) # Выведется: True
```

Значения булева типа можно привести к числовому типу с помощью функции `int()`: `True` будет преобразовано в `1`, `False` — в `0`.

Числа можно привести к типу bool с помощью функции `bool()`: `0` преобразуется в `False`, остальные числа — в `True`.

```
bool_true = True
bool_false = False
int_null = 0
int_one = 1

print (int(bool_true)) # Вывод в терминал: 1
print (int(bool_false)) # Вывод в терминал: 0
print (bool(int_null)) # Вывод в терминал: False
print (bool(int_one)) # Вывод в терминал: True
```