IT CAREER HUB

Урок 4.

Логический тип данных и сравнения

IT CAREER HUB

Логический тип данных

Логический тип данных (bool, от англ. *boolean*) используется для представления двух возможных значений: True (истина) и False (ложь).

Эти значения помогают программе принимать решения. Например, в зависимости от того, истинное значение или ложное, программа может выполнить разные действия.

Логические значения могут быть созданы напрямую или быть результатом логических операций, сравнений или приведения типов к **bool**.

Создание напрямую:

```python
isFinished = True
hasAccess = False
```

IT CAREER HUB

Приведение типов к bool

В Python существуют значения, которые интерпретируются как True или False при преобразовании в логический тип:

- **True: Любое ненулевое число, непустая строка, список, кортеж и т.д.**
- **False: None, 0, пустые последовательности (например, [], {}, "").**

Существует два способа преобразования значений в логический тип данных bool: **автоматическое** и **явное (вручную)**.

1. **Автоматическое преобразование**:
   Python автоматически преобразует значения в True или False, когда это необходимо, например, в условиях if, циклах и логических операциях. При этом Python определяет, является ли значение **"истинным"** или **"ложным"**, исходя из его типа.
2. **Явное (ручное) преобразование**:
   Для явного преобразования значений в логический тип данных можно использовать функцию **bool**.

```python
print(bool(5))          # True (ненулевое число)
print(bool(0))          # False
print(bool(None))       # False
print(bool([]))         # False (пустой список)
print(bool("Hello"))    # True (непустая строка)
```

IT CAREER HUB

Задание для закрепления 1

Определите, какие из этих значений преобразуются в `True` при использовании функции `bool()`.

a. `42`
b. `0`
c. `-2`
d. `""`
e. `"0"`
f. `"Python"`
g. `"False"`
h. `[]`
i. `None`
j. `3.14`

Посмотреть ответ

IT CAREER HUB

Операторы сравнения

Операторы сравнения в Python используются для сравнения двух значений. Результатом операции всегда является логическое значение — True или False. Эти операторы применяются для проверки равенства, неравенства, а также для сравнения величин (больше, меньше и т.д.).

Основные операторы сравнения:

Оператор	Описание	Пример	Результат
==	Равно	5 == 5	True
!=	Не равно	5 != 5	False
>	Больше	3 > 5	False
<	Меньше	4 < 6	True
>=	Больше или равно	5 >= 5	True
<=	Меньше или равно	3 <= 2	False

IT CAREER HUB

Примеры использования операторов сравнения с числами

1. **Оператор** == **(равно)**: Сравнивает два числа на равенство.

Пример

```python
a = 10
b = 10
print(a == b)  # True, так как a равно b
```

2. **Оператор** != **(не равно)**: Проверяет, не равны ли два числа.

Пример

```python
a = 10
b = 5
print(a != b)  # True, так как a не равно b
```

3. **Оператор** > **(больше)**: Проверяет, больше ли одно число другого.

Пример

```python
a = 8
b = 10
print(a > b)  # False, так как a меньше b
```

4. **Оператор** < **(меньше)**: Проверяет, меньше ли одно число другого.

Пример

```python
a = 5
b = 7
print(a < b)  # True, так как a меньше b
```

5. **Оператор** >= **(больше или равно)**: Проверяет, больше или равно ли одно число другому.

Пример

```python
a = 5
b = 5
print(a >= b)  # True, так как a равно b
```

6. **Оператор** <= **(меньше или равно)**: Проверяет, меньше или равно ли одно число другому.

Пример

```python
a = 3
b = 4
print(a <= b)  # True, так как a меньше b
```

IT CAREER HUB

Задание для закрепления 2

Соотнесите оператор сравнения с его результатом:

1. 5 != 3	a. False
2. 7 < 6	b. True
3. 8 >= 8	
4. 4 > 2	

Посмотреть ответ

IT CAREER HUB

Логические операторы

Логические операторы используются для выполнения логических операций над значениями или выражениями. Они возвращают True или False в зависимости от условия и позволяют комбинировать несколько условий в одном выражении.

Оператор	Описание	Пример	Результат
and	Логическое "И": возвращает True, если оба условия истинны	True and False	False
or	Логическое "ИЛИ": возвращает True, если хотя бы одно условие истинно	True or False	True
not	Логическое "НЕ": возвращает противоположное значение	not True	False

IT CAREER HUB

Оператор and (логическое "И")

Оператор and возвращает `True`, если **оба** условия истинны. Если хотя бы одно из условий ложно, результатом будет `False`.

Пример

```python
a = 5
b = 10

# Проверяем что оба числа положительные
print(a > 0 and b > 0)     # True, так как оба условия истинны
```

Пример из жизни для логического оператора and

Представьте, что вы хотите пойти на прогулку, но только если **солнечно** и у вас есть **свободное время**. Оба условия должны быть истинными, чтобы вы пошли гулять.

- **Возможные ситуации:**
  - **Солнечно** `True` и **свободное время** `True` → Идёте гулять - `True`.
  - **Солнечно** `False` и **свободное время** `True` → Не идёте гулять - `False`.
  - **Солнечно** `True` и **свободное время** `False` → Не идёте гулять - `False`.
  - **Солнечно** `False` и **свободное время** `False` → Не идёте гулять - `False`.

Это как сказать: "Я пойду гулять, если одновременно и погода хорошая, и у меня есть свободное время."

IT CAREER HUB

Оператор or (логическое "ИЛИ")

Оператор or возвращает `True`, если хотя бы **одно** из условий истинно. Если оба условия ложны, результатом будет `False`.

Пример

```python
a = -5
b = 10
# Проверяем что хотя бы одно из чисел положительное
print(a > 0 or b > 0)     # True, так как b > 0 истинно
```

Пример из жизни для логического оператора or

Теперь представьте, что вы хотите заказать пиццу, но вы сделаете заказ, если у вас либо **голод**, либо **желание пиццы**. В этом случае, достаточно, чтобы хотя бы одно из условий было истинным.

- **Возможные ситуации:**
  - **Голод** True и **желание пиццы** True → Закажете пиццу - True.
  - **Голод** False и **желание пиццы** True → Всё равно закажете пиццу, потому что хотите пиццу - True.
  - **Голод** True и **желание пиццы** False → Закажете, потому что голодны - True.
  - **Голод** False и **желание пиццы** False → Не закажете пиццу, так как оба условия ложны - False.

Это как сказать: "Я закажу пиццу, если либо я голоден, либо у меня есть настроение её съесть."

IT CAREER HUB

Оператор not (логическое "НЕ")

Оператор not возвращает **противоположное** значение. Если выражение истинно, not превращает его в ложное, и наоборот.

Пример

```python
a = False
print(not a)  # True, так как это обратное значение от False
```

Пример из жизни для логического оператора or

Представьте, что вы на работе и планируете уйти домой. Вы пойдёте домой только в том случае, если у вас нет задач. Здесь оператор not работает как отрицание: если задач нет, то вы идёте домой.

- Если у вас нет задач: not → "Иду домой" - True.
- Если задачи есть: not превращает это в - False, т.е. "Остаюсь на работе".

Это как сказать: "Я иду домой если у меня нет задач", что является логическим отрицанием выполнения работы.

IT CAREER HUB

Задание для закрепления 3

1. Соотнесите логический оператор с его результатом:

1. True and False	a. True
2. True or False	b. False
3. not True	

Посмотреть ответ

2. Какой результат выведет следующий код?

```python
x = 5
print(not x > 3)
```

a. True
b. False
c. Ошибка
d. None

Посмотреть ответ

3. Какой результат выведет следующий код?

```python
a = 5
b = 10
print(a < b and b < 10)
```

a. True
b. False
c. Ошибка
d. None

Посмотреть ответ

Комбинирование логических операторов

Вы можете комбинировать несколько логических операторов в одном выражении для создания более сложных условий.

Пример

```python
a = 5
b = 10
print(a > 0 and b > 0 or a == 5)
```

В этом примере результат будет True, потому что оба числа больше 0 (условие с and истинно), и также a == 5 (условие с or истинно).

IT CAREER HUB

Приоритет логических операторов

В Python логические операторы имеют определённый порядок выполнения (приоритет). Это значит, что некоторые операторы выполняются раньше других, если они встречаются в одном выражении.

Порядок приоритета логических операторов:

1. not (логическое "НЕ") — выполняется первым.
2. and (логическое "И") — выполняется вторым.
3. or (логическое "ИЛИ") — выполняется последним.

Пример

```python
a = True
b = True
c = True

print(not a or b and c)
```

В этом выражении сначала выполнится оператор not, затем and, и последним — or:

1. not a or b and c # Подставим значения переменных
2. not True or True and True # not True → False
3. False or True and True # True and True → True
4. False or True # False or True → True
5. True

Если вы хотите изменить порядок выполнения логических операторов, можно использовать скобки, чтобы явно указать, какие операции должны быть выполнены в первую очередь, так как скобки имеют наивысший приоритет.

IT CAREER HUB

Пример

```python
result = not (a or b) and c
```

Теперь оператор or выполняется первым (из-за скобок), затем not, и в последнюю очередь — and:

1. not (a or b) and c # Подставим значения переменных
2. not **(False or True)** and True # False or True → True
3. **not True** and True # not True → False
4. **False and True** # False and True → False
5. **False**

Приоритет математических операций и операторов сравнения

Когда в Python выполняются математические операции вместе с операторами сравнения, важно понимать, как Python решает, что выполнять в первую очередь. **Математические операции** имеют более высокий приоритет, чем **операторы сравнения**. Это значит, что сначала Python выполнит все вычисления, а затем сравнит результат.

Пример

```python
result = 3 + 2 > 4  # Сначала выполняется сложение (3 + 2), затем результат сравнивается с 4
print(result)       # True, так как 5 > 4
```

Пример из жизни для нескольких условий

1. **Использование and и or**

Предположим, вы хотите узнать, можно ли пойти гулять. Условия: (погода хорошая **или** день выходной), **и** у вас есть свободное время.

```python
good_weather = True
is_weekend = False
free_time = True

can_go_out = (good_weather or is_weekend) and free_time
print(can_go_out)  # Результат: True
```

2. **Использование not с другими операторами**

Допустим, вы планируете поехать в отпуск, если у вас **нет** работы **и** достаточно денег.

```python
have_work = True
enough_money = True

can_go_vacation = not have_work and enough_money
print(can_go_vacation)  # Результат: False
```

Пояснение:

- not have_work → превращает True в False (у вас есть работа, значит, вы не можете поехать).
- and enough_money → проверяет, достаточно ли денег (результат: False, так как денег недостаточно).
- Окончательный результат: False, вы не можете поехать в отпуск.

IT CAREER HUB

Таблица истинности

Таблица истинности показывает результат работы логических операторов для всех возможных комбинаций значений True и False.

Таблица истинности логических операторов в Python:

A	B	A and B	A or B	not A
True	True	True	True	False
True	False	False	True	False
False	True	False	True	True
False	False	False	False	True

IT CAREER HUB

Альтернативная интерпретация логических операторов

В Python логические операторы and, or и not можно сопоставить с арифметическими операциями. Оператор and похож на умножение (*), а or — на сложение (+), not на инверсию.

Таблица истинности с аналогией для всех трёх операторов:

A	B	A and B (A * B)	A or B (A + B)	not A (инверсия A)
True	True	True (1 * 1)	True (1 + 1)	False (1 → 0)
True	False	False (1 * 0)	True (1 + 0)	False (1 → 0)
False	True	False (0 * 1)	True (0 + 1)	True (0 → 1)
False	False	False (0 * 0)	False (0 + 0)	True (0 → 1)

IT CAREER HUB

Двойные неравенства

В Python можно использовать **двойные неравенства**, что позволяет проверять, попадает ли число в определённый диапазон значений. Это делает код более читабельным и удобным, поскольку не требует явного использования логических операторов and.

Синтаксис двойных неравенств:

```python
min_value < variable < max_value
```

Этот синтаксис позволяет проверить, находится ли переменная между двумя значениями.

Пример

```python
x = 5
print(1 < x <= 10) # x находится между 1 и 10 (включительно)
```

В этом примере Python проверяет сразу два условия:

1. 1 < x — проверка, что x больше 1.
2. x <= 10 — проверка, что x меньше или равно 10.

Эквивалент без двойного неравенства:

Использование двойных неравенств эквивалентно следующему коду с логическим оператором and:

```python
x = 5
print(1 < x and x <= 10) # x находится между 1 и 10 (включительно)
```

Оба подхода работают одинаково, но двойное неравенство делает код более кратким и читаемым.

IT CAREER HUB

Задание для закрепления 4

1. Что произойдёт при выполнении следующего кода?

```python
x = 3
print(1 < x < 5)
```

a. True
b. False
c. Ошибка
d. None

Посмотреть ответ

2. Какой результат выведет следующий код?

```python
a = False
b = True
c = True
print(not a or b and c)
```

a. True
b. False
c. Ошибка
d. None

Посмотреть ответ

IT CAREER HUB

Ответы на задания

Задания на закрепление 1	Вернуться к заданиям
`True` значения	Ответ: a, c, e, f, g, j
Задания на закрепление 2	Вернуться к заданиям
`True/False`	Ответ: 1 - b, 2 - a, 3 - b, 4 - b
Задания на закрепление 3	Вернуться к заданиям
1. `True/False`	Ответ: 1 - b, 2 - a, 3 - b
2. Результат выполнения кода	Ответ: b
3. Результат выполнения кода	Ответ: b
Задания на закрепление 4	Вернуться к заданиям
1. Результат выполнения кода	Ответ: a
2. Результат выполнения кода	Ответ: a

IT CAREER HUB

Практическая работа

1. Число в диапазоне

Напишите программу, которая получит два числа от пользователя и выведет, попадает ли первое число в диапазон от 1 до второго числа (включая границы).

Пример вывода:

```
Введите первое число: 3
Введите второе число: 5
True
```

Решение:

```python
a = 7
b = 3

print("Результат операции 'Разность':", a - b)
print("Результат операции 'Остаток от деления':", a % b)
print("Результат операции 'Целочисленное деление':", a // b)
```

2. Сравнение чисел

Напишите программу, которая получит от пользователя три числа и выведет результат сравнения первого числа с остальными двумя, используя операторы `больше`, `меньше` и `равны`.

Пример вывода:

```
Введите первое число: 7
Введите второе число: 5
Введите третье число: 7
Первое больше второго: True
Первое меньше третьего: False
Первое равно третьему: True
```

IT CAREER HUB

Решение:

```python
price = int(input("Введите стоимость товара: "))
quantity = int(input("Введите количество товара: "))
payment = int(input("Введите сумму оплаты: "))

total_cost = price * quantity
change = payment - total_cost

print("Сдача:", change, "рублей")
```