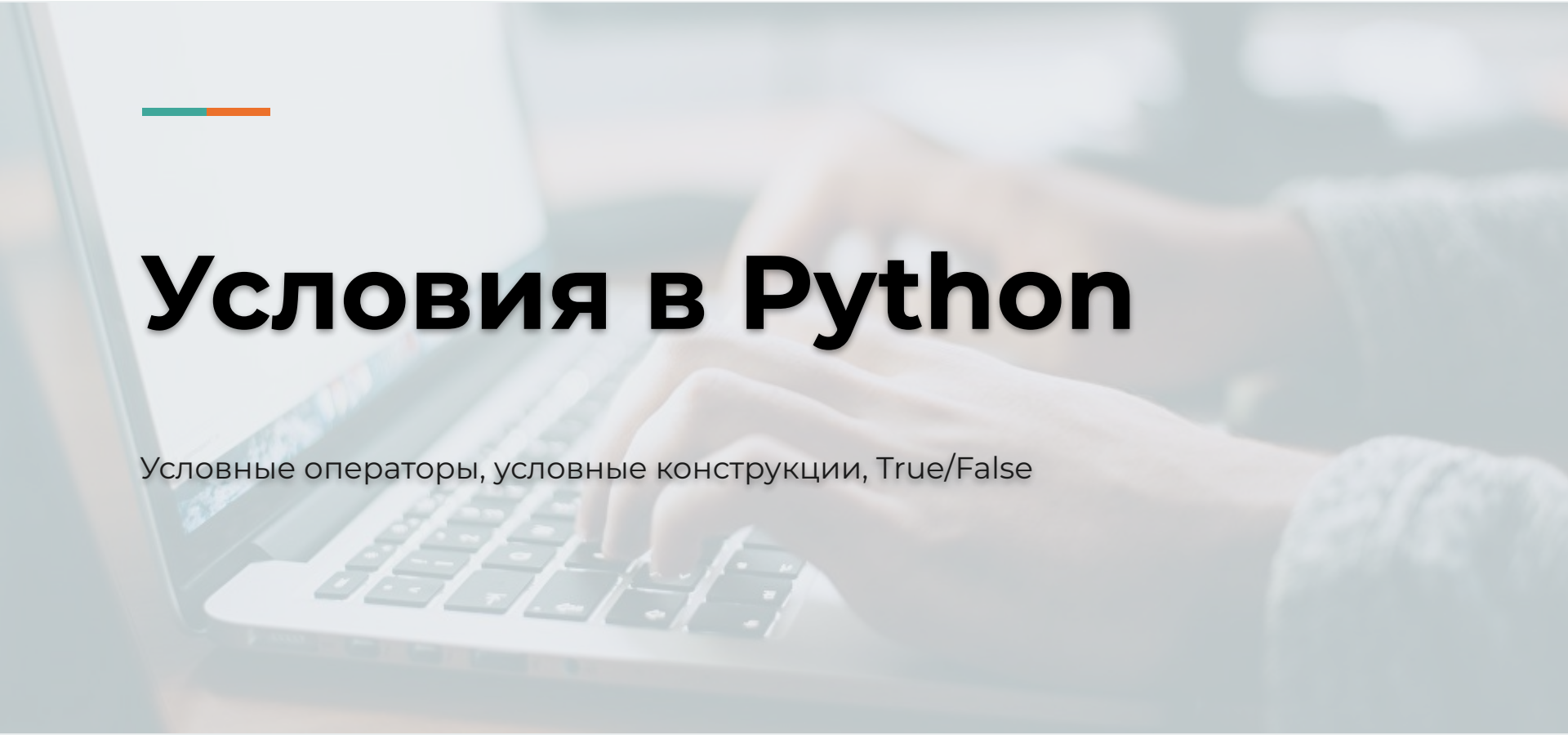

# Условия в Python

Условные операторы, условные конструкции, True/False

# Условия

## Операторы сравнения

# Операторы сравнения

В прошлых темах уже встречались арифметические операторы в Python. Кроме них существуют и операторы сравнения. Они сравнивают значение слева и справа, и в зависимости от результата сравнения возвращают значение типа bool (логические значения True или False).

# Операторы сравнения

1. `==` Проверяет, равны ли значения слева и справа.
Если равны, то условие становится истинным и возвращает True, если не равны, то возвращает False

2. `!=` Проверяет, НЕ равны ли значения слева и справа.
Если значения не равны, то условие становится истинным и возвращает True, если они равны - возвращает False

# Операторы сравнения

3 **>** Проверяет, больше ли значение слева, чем значение справа.
Если больше, то условие становится истинным и возвращает True, иначе возвращает False.

4 **<** Проверяет, меньше ли значение слева, чем значение справа.
Если меньше, то условие становится истинным и возвращает True, иначе возвращает False.

# Операторы сравнения

5 **>=** Проверяет, больше или равно значение слева, чем значение справа.
Если так, то условие становится истинным и возвращает True, иначе возвращает False.

6 **<=** Проверяет, меньше или равно значение слева, чем значение справа.
Если так, то условие становится истинным и возвращает True, иначе возвращает False.

# Следующий код нужно печатать в Python Shell.

## ЗАДАНИЕ

Выполните следующие сравнения и посмотрите, каким будет результат.

```
10 == 8
11.5 < 11.5
12 > 15
```

```
10 != 8
11.5 <= 11.5
15 >= 12
```

# Условия

Синтаксис инструкции if-elif-else

# Синтаксис инструкции if

Условная инструкция состоит как минимум из одного элемента – это if. В начале строки пишется ключевое слово if, после чего записывается условие. Не забываем ставить двоеточие в конце! Строкой ниже пишется то действие, которое должно выполниться если условие истинное и вернет True.

```
if условие:
----действие1
действие2
```

# Синтаксис инструкции if

Важно: те действия, которые должны выполняться, если условие верное, пишутся с четырьмя пробелами в начале. Так Python понимает, что относится к условию, а что нет. В примере действие1 выполнится только если условие истинно, но действие2 выполнится в любом случае.

```
if условие:
----действие1
действие2
```

# Синтаксис инструкции if

```
x = int(input())
if x == 0:
    print("Вы ввели 0")
print("Программа завершилась")
```

Условное выражение

Код написан с четырьмя пробелами, значит, находится в теле условия и выполнится, если x будет равен нулю.

Выполнится в любом случае.

# Синтаксис инструкции if

В качестве примера разберите этот код. Запустите его несколько раз, вводя разные числа и сравните результаты.

```
x = int(input())
if x == 0:
    print("Это ноль")
if x > 0:
    print("Это положительное  число")
if x < 0:
    print("Это отрицательное  число")
if x != 0:
    print("Это не ноль")
```

# Синтаксис инструкции if - else

Бывают случаи, когда необходимо предусмотреть альтернативный вариант выполнения программы. То есть при истинном условии нужно выполнить одно действие, при ложном – другое. Для этого используется конструкция if – else.

```
if условие:
    действие1
else:
    действие2
```

# Синтаксис инструкции if - else

```
x = int(input())
if x == 0:
    print("Вы ввели 0")
else:
    print("Вы ввели не 0")
print("Программа завершилась")
```

Условное выражение

Код выполнится если x будет равен нулю.

Код выполнится во всех остальных случаях.

Выполнится в любом случае.

## ЗАДАНИЕ

Выполните следующие сравнения и посмотрите, каким будет результат.

```
a = 10
if a % 2 == 0:
        print(“yes”)
else:
        print(“no”)
```

# Синтаксис инструкции if - elif - else

Еще один вариант условных инструкций – это if-elif-else. Действие2 выполнится, если условие1 будет ложным, а условие2 – истинным. Если и условие1 и условие2 будут ложными, то выполнится действие3

```
if условие1:
    действие1
elif условие2:
    действие2
else:
    действие3
```

# Синтаксис инструкции if - elif - else

```
x = int(input())
if x == 0:
    print("Вы ввели 0")
elif x > 0:
    print("Вы ввели больше, чем 0")
else:
    print("Вы ввели меньше, чем 0")

print("Программа завершилась")
```

Первое условное выражение

Код выполнится если x будет равен нулю.

Второе условное выражение

Код выполнится если не выполняется первое условие и x больше нуля.

Код выполнится во всех остальных случаях.

Выполнится в любом случае.

## ЗАДАНИЕ

Выполните следующие сравнения и посмотрите, каким будет результат при a равном 10, -666, 0.

```
if a == 0:
    print(“yes”)
elif a >= 0:
    print(“+”)
else:
    print(“-”)
```

# Условия

Условный тернарный оператор

Логические операторы

Оператор in

# Условный тернарный оператор

Иногда в проектах есть необходимость создания коротких условий, но тем не менее они занимают минимум четыре строки. Для таких ситуаций существует короткая форма записи условий. На примере ниже показаны два одинаковых условия, но форма записи разная.

```
x = 10 if x < 0 else -10
```

```
if x < 0:
    x = 10
else:
    x = -10
```

## ЗАДАНИЕ

Условие ниже представьте с использованием условного тернарного оператора.

```
if a % 2 == 0:
    print (b * 2)
else:
    print (b / 2)
```

# Логические операторы

Иногда нужно проверить одновременно не одно, а несколько условий. Для этого в Python существуют стандартные логические операторы: логическое И (and), логическое ИЛИ (or), логическое отрицание НЕ (not).

# Логические операторы

Логическое **И** (and) возвращает True тогда и только тогда, когда оба его операнда имеют значение True.

Логическое **ИЛИ** (or) возвращает True тогда и только тогда, когда хотя бы один операнд равен True.

Логическое **НЕ** (not) возвращает True, если операнд равен False и наоборот.

# Логические операторы

Действие1 произойдет, если **И** условие1 **И** условие2 будут верны.

Действие2 выполнится, если **ИЛИ** условие3 верное **ИЛИ** условие4 верное **ИЛИ** оба одновременно. Т.е. хотя бы одного верного условия достаточно.

Действие3 выполнится, если условие5 будет **НЕ**верное.

```
if условие1 and условие2:
    действие1
if условие3 or условие4:
    действие2
if not условие5:
    действие3
```

# Оператор in

**in** проверяет, входит ли элемент слева в состав последовательности справа. Если да, то условие становится истинным и возвращает True, иначе возвращает False.

```
>>> "a" in "Banana"
True
>>> 2 in [1, 2, 3, 4]
True
```

## ЗАДАНИЕ

Дано число.

- Если оно делится на 2, на 3, и на 5, то вывести фразу “2, 3 и 5”;
- Если делится только на 2 и 3, то “2 и 3”;
- Если делится только на 2, то “2”;
- Но если число не делится ни на 2, ни на 3, ни на 5, то вывести “0”.

В этой задаче нельзя пользоваться else.

# Условия

Вложенные условные инструкции

# Вложенные условные инструкции

Внутри условных инструкций можно использовать любые инструкции, в том числе и еще одну условную инструкцию. Получаем вложенное ветвление – после одной развилки в ходе исполнения программы появляется другая развилка. При этом вложенные блоки имеют больший размер отступа (например, 8 пробелов).

```
if условие1:
----действие1
----if условие2:
--------действие2
----else:
--------действие3
else:
----действие4
```

# Вложенные условные инструкции

Рассмотрите внимательнее этот пример, напишите такую же программу и запустите. Все ли понятно? Если нет – зовите тренера.

```
x = int(input())
if x > 0:
    print("Это не ноль")
    if x % 2 == 0:
        print("Это положительное четное число")
    else:
        print("Это положительное нечетное число")
elif x < 0:
    print("Это не ноль")
    if x % 2 == 0:
        print("Это отрицательное четное число")
    else:
        print("Это отрицательное нечетное число")
else:
    print("Это ноль!")
```

4 пробела

8 пробелов