



# MACHINE LEARNING

ТЕХНОЛОГИЯ ПРОГРАММИРОВАНИЯ НА ЯЗЫКЕ PYTHON И  
РАЗРАБОТКА ПРОГРАММ ДЛЯ МАШИННОГО ОБУЧЕНИЯ  
ЛЕКЦИЯ II

# План занятия

- Программа. Язык программирования. Трансляция
- Интерактивный режим
- Типы данных
- Определение переменной
- Логические выражения
- Условный оператор
- Инструкция if, if-else, if-elif-else
- Множественное ветвление

# Трансляторы и компиляторы

**Транслятор** – специальная программа, преобразующая программный код с того или иного языка программирования в машинный код

**Компилятор**  
Сразу переводит весь программный код на машинный язык.  
Создает исполняемый файл.

**Интерпретатор**  
Переводит программный код построчно.  
Напрямую взаимодействует с операционной системой.

# Данные и их типы

- Специальные типы: None, NotImplemented и Ellipsis ;
- Числа:
  - целые
    - обычное целое int
    - целое произвольной точности long
    - логический bool
  - число с плавающей точкой float
  - комплексное число complex
- Последовательности:
  - неизменяемые:
    - строка str
    - Unicode-строка Unicode
    - кортеж tuple
  - изменяемые:
    - список list
- Отображения:
  - словарь dict

# Данные и их типы

- Объекты, которые можно вызвать:
  - функции (пользовательские и встроенные)
  - функции-генераторы;
  - методы (пользовательские и встроенные);
  - классы (новые и "классические");
  - экземпляры классов (если имеют метод `__call__`);
  - модули;
  - классы (см. выше);
  - экземпляры классов (см. выше);
  - файлы `file` ;
  - вспомогательные типы `buffer`, `slice`.

Узнать тип любого объекта можно с помощью встроенной функции `type()`

# Целые числа (integer)

Целые числа (integer) – положительные и отрицательные целые числа, а также 0 (например, 4, 687, -45, 0).

Значения типа `int` должны покрывать диапазон от `-2147483648` до `2147483647`, а точность целых произвольной точности зависит от объема доступной памяти.

# Основные операции с целыми числами

- $A + B$  — сумма;
- $A - B$  — разность;
- $A * B$  — произведение;
- $A / B$  — частное, *(результатом этого действия является вещественное число, даже если  $A$  нацело делится на  $B$ );*
- $A \% B$  — взятие остатка от деления  $A$  на  $B$ ;
- $A // B$  — взятие целой части от деления  $A$  на  $B$ ;
- $A ** B$  — возведение в степень.

# Приоритеты операций

Приоритеты операций в Python совпадают с приоритетом операций в математике, а именно:

- Выполняются возведения в степень справа налево, то есть  $3 ** 3 ** 3$  это  $3 ** (3 ** 3)$ .
- Выполняются унарные минусы (отрицания).
- Выполняются умножения и деления слева направо. Операции умножения и деления имеют одинаковый приоритет.
- Выполняются сложения и вычитания слева направо. Операции сложения и вычитания имеют одинаковый приоритет.
- Для изменения порядка действий нужно использовать скобки.



# Числа с плавающей точкой (float point)

Числа с плавающей точкой (float point) – дробные числа (например, 1.45, -3.789654, 0.00453). Примечание: разделителем целой и дробной части служит точка, а не запятая.

# Строки (string)

Строки (`string`) — набор символов, заключенных в кавычки (например, `"ball"`, `"What is your name?"`, `'dkfjUUv'`, `'6589'`). Примечание: кавычки в Python могут быть одинарными или двойными.

# Основные операции со строками

- $A + B$  — конкатенация (строка  $B$  приписывается к строке  $A$ );
- $A * n$  — повторение  $n$  раз, значение  $n$  должно быть целого типа.

# Изменение типа данных

Функция `int()` преобразует переданную ей строку (или число с плавающей точкой) в целое, функция `str()` преобразует переданный ей аргумент в строку, `float()` - в дробное число.

Выражение	Результат
<code>int ("56")</code>	выполнения
<code>int (4.03)</code>	56
<code>int ("comp 486")</code>	4
<code>str (56)</code>	Ошибка
<code>str (4.03)</code>	'56'
<code>float (56)</code>	'4.03'
<code>float ("56")</code>	56.0
	56.0

# Переменные в Python

Переменная – это ссылка на область памяти, где хранятся те или иные данные.



# Имена переменных

- Имя переменной может состоять только из цифр, букв и символов подчеркивания
- Имя переменной не может начинаться с цифр
- Имя должно описывать суть, т.е. нужно давать имена, говорящие о назначении данных, на которые они ссылаются
- Имя переменной не должно совпадать с командами языка (зарезервированными ключевыми словами)
- Имя переменной принято начинать со строчной буквы
- Не следует создавать имена длиннее 15 символов

# Ввод и вывод данных

Ввод и вывод данных осуществляется с помощью встроенных функций:

- Ввод: `input(параметры)`
- Вывод: `print(параметры)`

# Ввод данных

## 1) Непосредственный

**ввод**

```
input()
```

**1234**

```
'1234'
```

```
input()
```

**Hello World!**

```
'Hello World!'
```

## 2) Параметр – приглашение

```
input('Введите число: ')
```

Введите число: **10**

```
'10'
```

```
int(input('Введите число: '))
```

Введите число: **10**

```
10
```

```
float(input('Введите число: '))
```

Введите число: **10**

```
10.0
```

## 3) Тип данных – строчный

```
input('Введите номер карты: ')
```

Введите номер карты: **123456**

```
'123456'
```

```
input('Введите имя: ')
```

Введите имя: **Иван**

```
'Иван'
```

## 4) Присвоение значения

**переменной**

```
name=input('Введите ваше имя: ')
```

Введите ваше имя: **Иван**

```
print(name)
```

```
'Иван'
```



# Вывод данных

## 1) Тип данных строчный

- `print("Игра 'Monopoly' 2.0")`  
Игра 'Monopoly' 2.0
- `print("Этот", "солнечный", "день")`  
Этот солнечный день
- `print("Самое",  
"яркое",  
"воспоминание")`  
Самое яркое воспоминание

## 2) Вывод переменных

- `a=1;`  
`b=2;`  
`print(a, '+', b, '=', a+b)`  
1+2=3

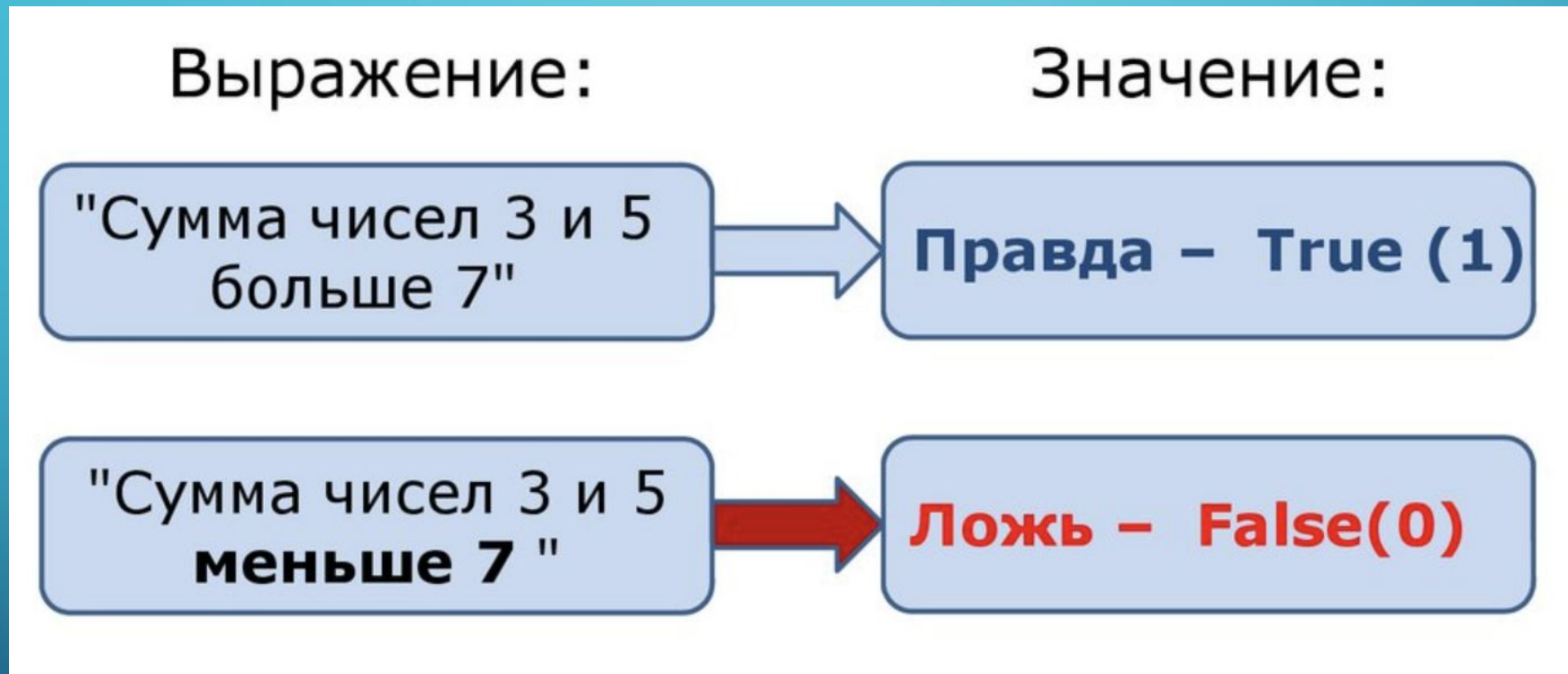
## 3) `sep` - параметр, используемый в качестве разделителя

- `a=1;`  
`b=2;`  
`c=a+b;`  
`print(a, b, c, sep = ':')`  
1:2:3

## 4) `end` – параметр, который указывает на то, что выводится после вывода всех значений, перечисленных в функции `print`

- `print(a, b, c, sep = ' ', end = '\n')`

# Логические выражения и логический тип данных



Если результатом вычисления выражения может быть лишь истина или ложь, то такое выражение называется логическим.

# Логические выражения и логический тип данных

## Операторы сравнения

### Операторы сравнения в Python

Оператор	Значение	Пример
<code>==</code>	равно	<code>x == y</code>
<code>!=</code>	не равно	<code>x != y</code>
<code>&gt;</code>	больше чем	<code>x &gt; y</code>
<code>&lt;</code>	меньше чем	<code>x &lt; y</code>
<code>&gt;=</code>	больше чем или равно	<code>x &gt;= y</code>
<code>&lt;=</code>	меньше чем или равно	<code>x &lt;= y</code>

# Примеры работы с логическими выражениями

```
x = 12 - 5 # это не логическая операция, а  
           операция присваивания переменной x  
           результата выражения 12 - 5  
  
x == 4    # x равен 4  
x == 7    # x равен 7  
x != 7    # x не равен 7  
x != 4    # x не равен 4  
x > 5     # x больше 5  
x < 5     # x меньше 5  
x >= 6    # x больше или равен 6  
x <= 6    # x меньше или равен 6
```

# Сложные логические выражения

## Логические операторы

### Логические операторы в Python:

Оператор	Описание	Примеры
<b>and</b>	Логический оператор "И". Условие будет истинным если оба операнда истина.	True and True равно True. True and False равно False. False and True равно False. False and False равно False.
<b>or</b>	Логический оператор "ИЛИ". Если хотя бы один из операндов истинный, то и все выражение будет истинным.	True or True равно True. True or False равно True. False or True равно True. False or False равно False.
<b>not</b>	Логический оператор "НЕ". Изменяет логическое значение операнда на противоположное.	not True равно False. not False равно True.

# Примеры работы с логическими операторами

`x = 8 y = 13`

`x == 8 and y < 15`    *# x равен 8 и y меньше 15*

`x > 8 and y < 15`    *# x больше 8 и y меньше 15*

`x != 0 or y > 15`    *# x не равен 0 или y больше 15*

`x < 0 or y > 15`    *# x меньше 0 или y больше 15*

# Условный оператор

## Инструкция if

Смысл неполного условного оператора: «Если <выполняется условие>, делать какие-то действия»

if **a** лог.оператор **b**:

    действие, выполняемое при результате True

Пример:

```
if n < 100: # если значение переменной n меньше 100, то ...  
    c = a**b # возвести значение переменной a в степень b,  
# результат присвоить c.
```

Внимание! Не забывайте про отступы, так как они очень важны и являются частью кода. В Python-сообществе принято делать 4 пробела.

# Условный оператор

## Инструкция if

Общая форма записи:

```
if <условие>:
```

```
    <действие 1>
```

```
    <действие 2>
```

и т.д.

Пример:

```
if x==y:
```

```
    z=x+y
```

```
    z=z*z
```

Схема программы  
с конструкцией **if**





# Условный оператор

## Конструкция if с условием else

Смысл полного условного оператора: «Если <выполняется условие>, делать какие-то действия, иначе (если условие не выполняется – False) делать другие действия»

if a логический оператор b:

    действие, выполняемое при результате True

else:

    действие, выполняемое при результате False

Пример:

```
a=50;
```

```
b=35;
```

```
if a+b>100:
```

```
    print('Yes');
```

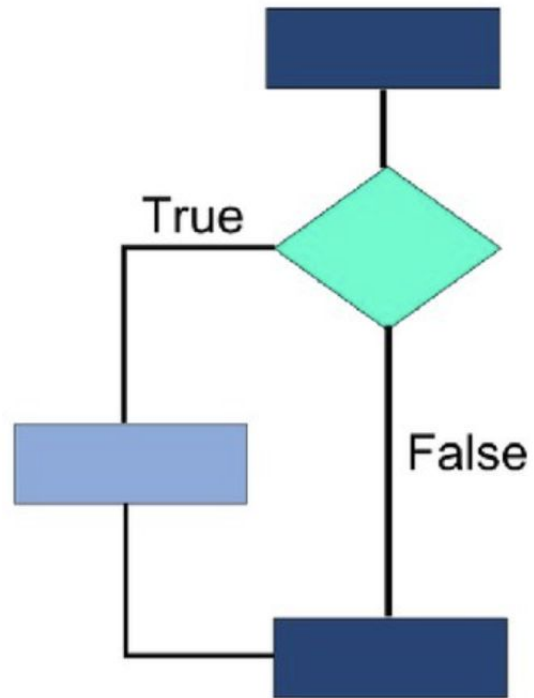
```
else:
```

```
    print('No')
```

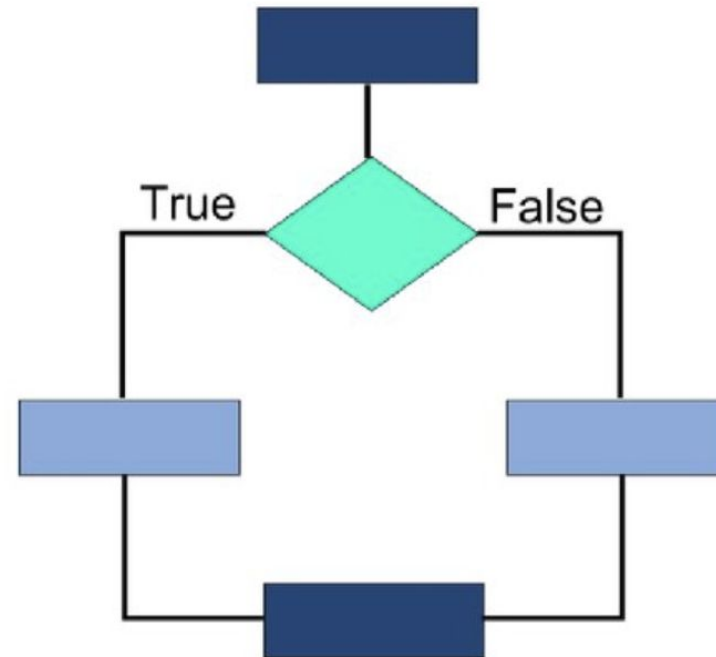
# Условный оператор

## Блок-схемы

Конструкция – **if**



Конструкция **if - else**



# Сложные условия

Пример:

- `a=50;`
- `b=35;`
- `if a>30 and b>100:`
- `print('Yes');`
- `else:`
- `print('No')`

Варианты усложнения условия:

`and` – “И”/”Логическое умножение”/Конъюнкция

`or` – “Или”/«Логическое сложение»/Дизъюнкция

`not` – ”Не”/Инверсия

Приоритет действий:

- 1) отношения (`<`, `>`, `<=`, `>=`, `==`, `!=`)
- 2) `not`
- 3) `and`
- 4) `or`

# Вложенные условные операторы

```
if a > b:  
    print("Андрей старше")  
else:  
    if a == b:  
        print("Одного возраста")  
    else:  
        print("Борис старше")
```

вложенный  
условный оператор

```
if <условие1> :  
    if <условие2> :  
        <оператор1>  
    else: <оператор2>  
else: <оператор3>
```

```
if логическое_выражение {  
    ... ;  
}  
else  
    if логическое_выражение {  
        ... ;  
    }  
    else  
        if логическое_выражение {  
            ... ;  
        }  
        else {  
            ... ;  
        }  
}
```

# Каскадное ветвление

## Конструкция if-elif-else

Если после else сразу следует еще один оператор if, можно использовать каскадное ветвление со служебным словом elif: если очередное условие ложно, выполняется проверка следующего условия и т.д.

Каскадное ветвление позволяет выбрать один из нескольких (а не только из двух) вариантов.

Примечание: elif = else if

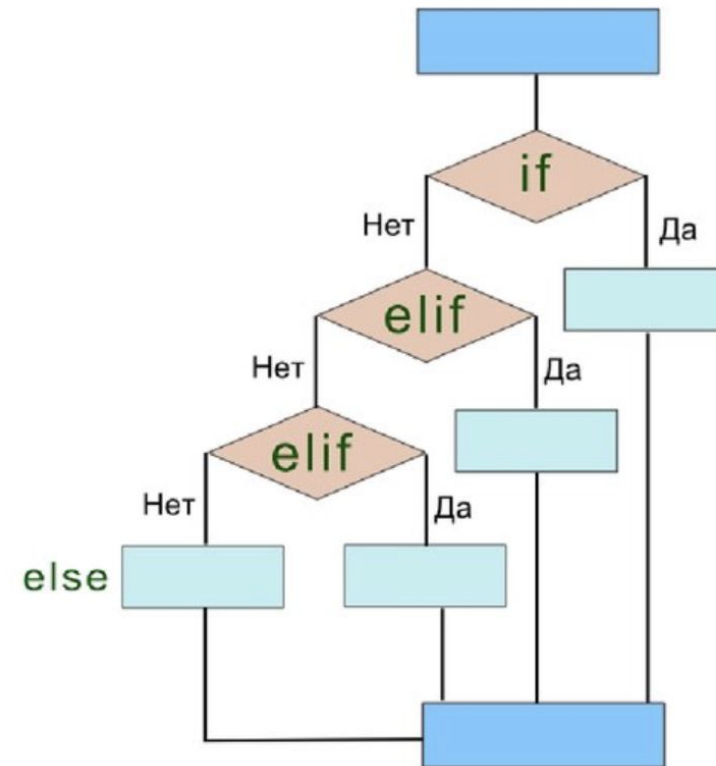
Пример:

```
If a>b:  
    print('Больше');  
  
elif a==b:  
    print('Равно');  
  
else:  
    print('Меньше')
```

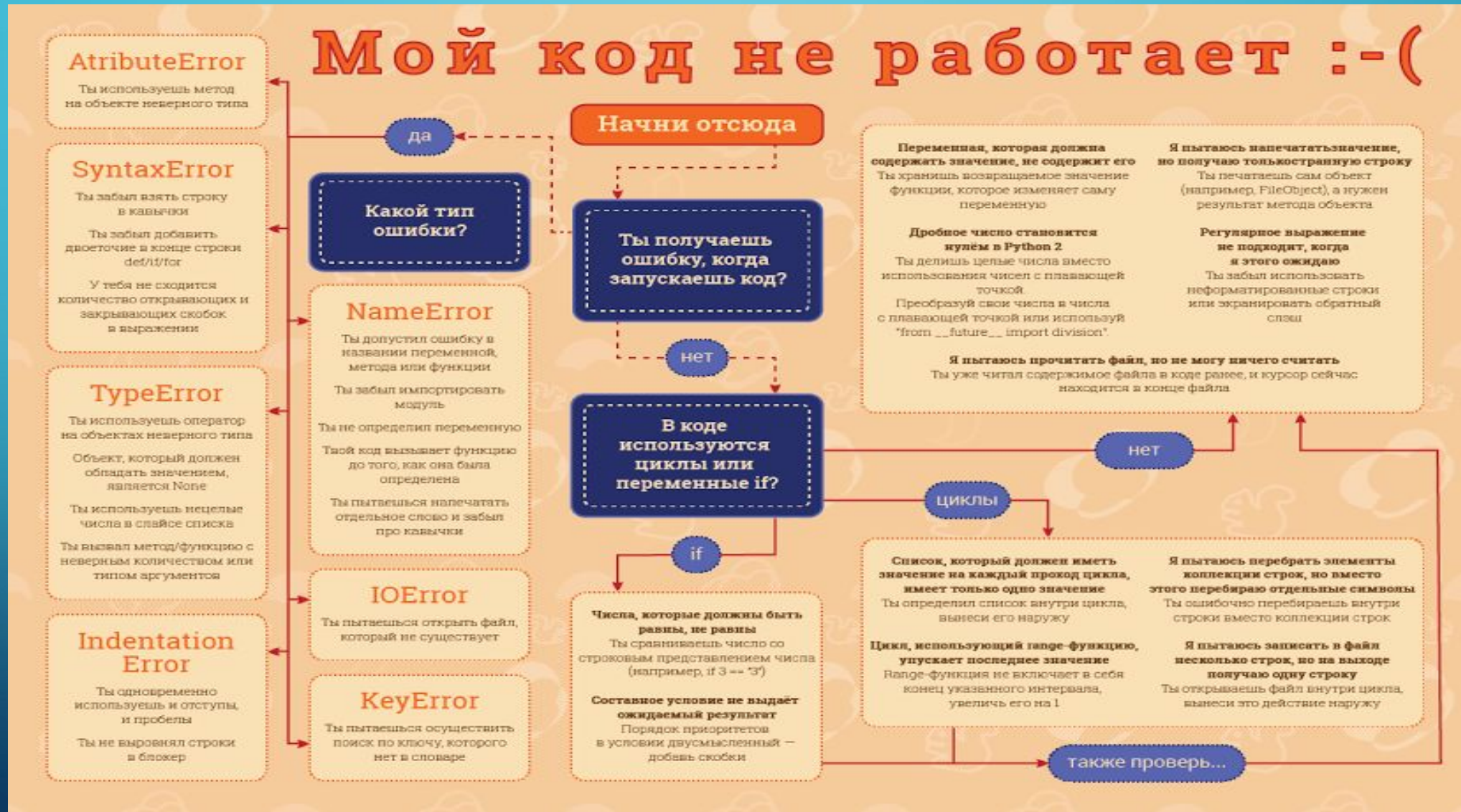
# Блок-схема конструкции if-elif-else

```
if <условие1>: <оператор 1>  
elif <условие2>: <оператор 2>  
...  
elif <условие n>: <оператор n>  
else : <оператор m>
```

Блок-схема **if - elif - else**



# Причины ошибок



# План следующего занятия

Библиотека Math

Циклы:

- while
- for

Строки:

- последовательности
- срезы
- индексация
- массивы
- кортежи
- специальные функции
- строковые методы



**Спасибо за внимание!**