

# DIRTY PYTHON 1.0

STONE 13TH PRESENTS

ЛЕКЦИЯ 7



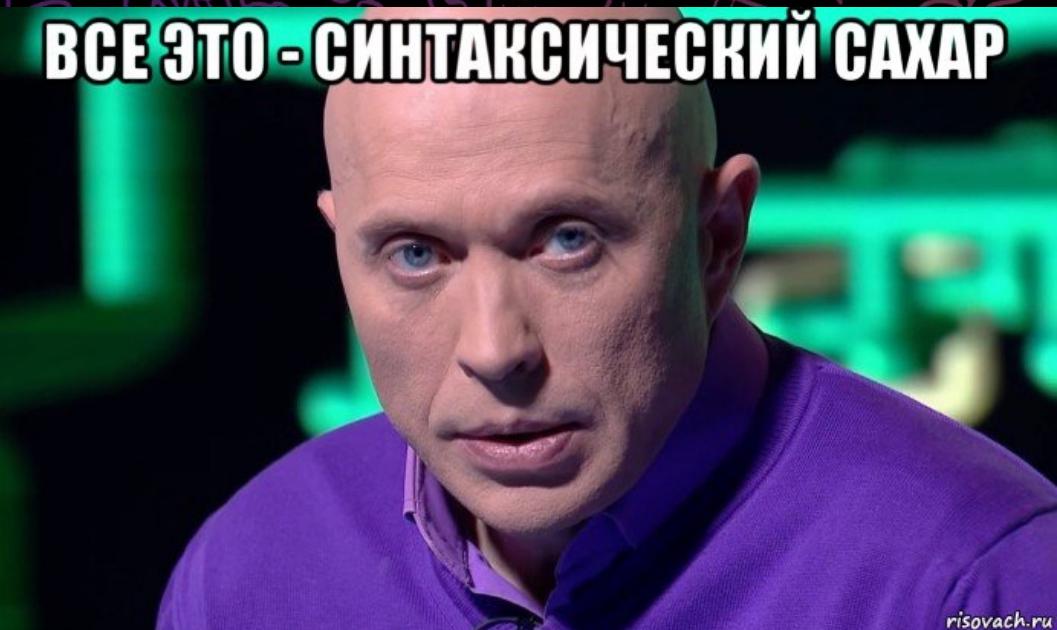
DIRTY  
PYTHON

# ЧТО У НАС СЕГОДНЯ?

Сахарок:

- Закрепим основные моменты
- Посмотрим на синтаксический сахар
- Всякие полезные фишки

ВСЕ ЭТО - СИНТАКСИЧЕСКИЙ САХАР

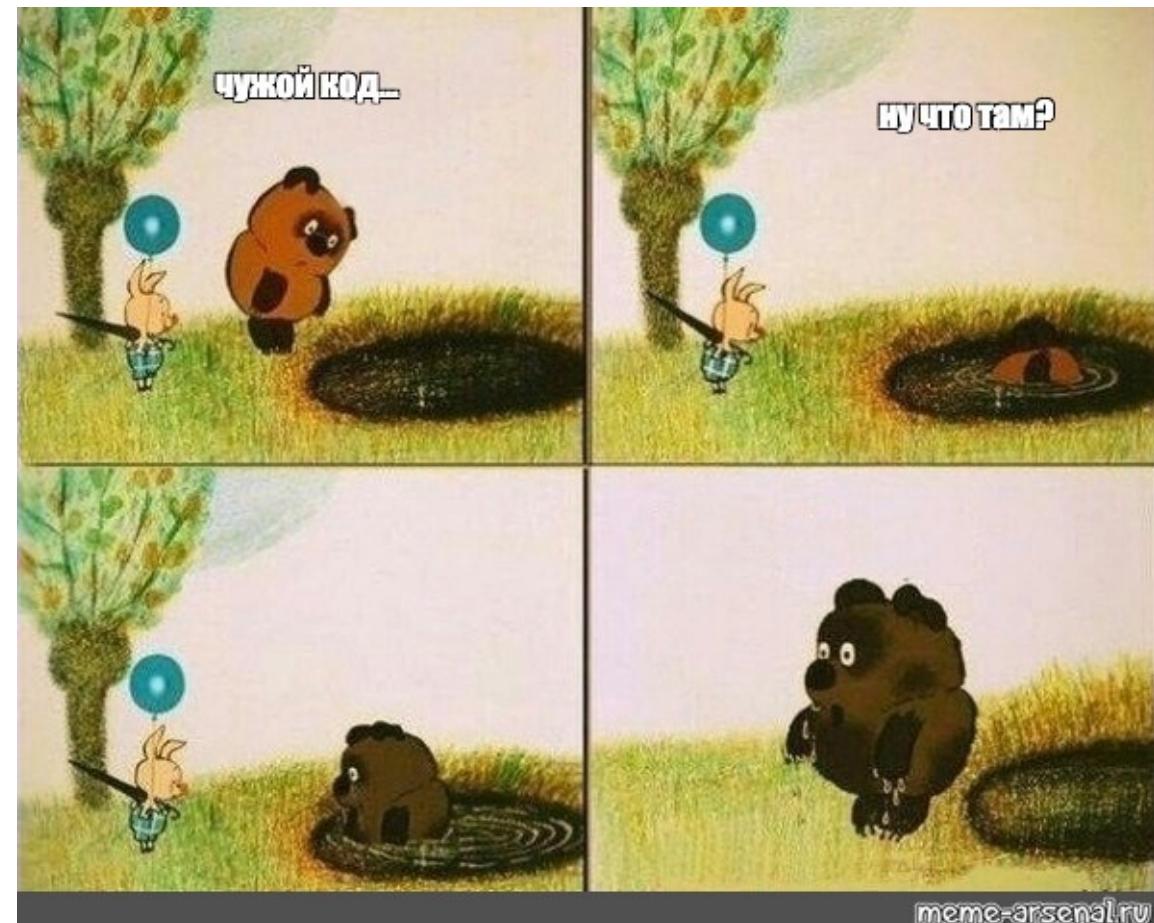


risovach.ru



# ЧТО ТАКОЕ СИНТАКСИЧЕСКИЙ САХАР? BIRTHDAY PYTHON

- Это способы сократить код и сделать его более красивым
  - Использовать не обязательно
  - Но хорошо знать, чтобы столкнувшись с чужим кодом, понимать откуда у него ноги растут и что вообще это значит



# ПЕРВАЯ ШТУКА: ОБМЕН ПЕРЕМЕННЫХ ЗНАЧЕНИЯМИ



- В одной строчке можно поменять значение переменных

a, b = b, a

- И не надо никакой промежуточной переменной
- И никто нас двумя переменными не ограничивает

a, b, c = c, a, b

- Главное, чтобы количество переменных справа и слева совпадало

The code editor shows the following Python script:

```
> 1
2     a = 5
3     b = 10
4     c = 15
5
6     a, b, c = c, a, b
7
8     print(f'a = {a}')
9     print(f'b = {b}')
10    print(f'c = {c}')


Run: lection7
```

The terminal window below shows the output of the script:

```
▶   ↑ /usr/local/bin/python3.1
▶   ↓ a = 15
▶   ↓ b = 5
▶   ↓ c = 10
```

# РАСПАКОВКА



- Можно распаковать список, кортеж и даже множество (но помним, что множество у нас не упорядочено!) сразу в переменные, главное чтобы количество элементов совпадало.
- Если мы не будем использовать в дальнейшем часть переменных, то можно использовать нижнее подчеркивание
- Если мы одной из переменных добавим \*, то в это переменную влезет все, что не достанется явно обозначенным переменным.
- Единственное, нельзя использовать две переменные со звездочками в одном присвоении – пайтон не поймет, сколько куда определять



# РАСПАКОВКА



```
> 13 my_list = [1, 2, 3, 4, 5]
14 a, b, c, d, e = my_list # каждой переменной присвоен элемент их списка
15 _, _, _, x, y = my_list # так мы присвоили значения двух последних элементов
16 *f, g, i = my_list      # в переменную f отправится список из первых трех элементов
17 j, *k = my_list         # в переменную k отправится список из всех элементов, кроме первого
18 l, *m, n = my_list      # присвоили первый и последний, остальные уйдут списком в m
19 print(a, b, c, d, e)
20 print(x, y,)
21 print(f, g, i,)
22 print(j, k,)
23 print(l, m, n)
24
```

```
run: lection7 ×
> ↗ /usr/local/bin/python3.10 /Users/di/Desktop/DirtyPython/dirtyPython1/Lection/lection7.py
↓ 1 2 3 4 5
→ 4 5|
→ [1, 2, 3] 4 5
→ 1 [2, 3, 4, 5]
→ 1 [2, 3, 4] 5
```



# МОРЖОВЫЙ ОПЕРАТОР

- Моржовый оператор одновременно присваивает и возвращает значения выражения
- Почему моржовый? Похож просто на моржа := с глазками и бивнями
- Используется обычно с if, иногда while
- Используется не часто, но бывает в однострочниках
- В целом знать неплохо, может пригодиться
- Вот тут больше примеров использования моржа

```
def func():
    return 5 # вот такая у нас функция: тупо возвращает 5
# ниже мы вызываем функцию и присваиваем ее возврат переменной a
if (a:= func()) > 3:
    print(f'{a} больше 3')
# как видите return функции у нас хранится в a
```

Run: lection7 ×  
/usr/local/bin/python3.10 /Users/di/Desktop/DirtyPython/dirtyPython1/Lection/lection7.py  
5 больше 3  
Process finished with exit code 0

<https://habr.com/ru/companies/otus/articles/555924/>

# LIST COMPREHENSION



- Уже говорили, но не грех и повторить
  - Формируем списки при помощи конструкции, которая заменяет использование цикла for для заполнения списка
  - Работает чуть быстрее, занимает меньше места да и просто выглядит красивее
  - Итак, еще раз про структуру `my_list = [i**2 for i in range(10) if i**2 % 2 == 0]`
- ✓ Используем квадратные скобки - у нас ведь список
- ✓ Перед for указываем что именно будем помещать в список: это может быть просто элемент или результат какого-то действия, выражения, функции
- ✓ Далее идет сам цикл for: обозначаем переменную и откуда мы ее берем – range или какой-то итерируемый объект (например строка, другой список и т.д.)
- ✓ Опционально можно добавить условие проверки if

По факту list comprehension может делать и map и filter одновременно, а потому и потеснил их из кода



```
numbers = []
for i in range(100):
    numbers.append(i)
```



```
numbers = [i for i in range(100)]
```

# LIST COMPREHENSION



- Если мы захотим строку '1234567890' превратить в список, который содержит только четные цифры (с типом int), то мы можем это сделать несколькими способами. Просто сравните код

```
34     my_list = '1234567890'
35     new_list = []
36     for s in my_list:
37         if int(s) % 2 == 0:
38             new_list.append(int(s))
39     print(new_list)
```

Run: lection7 ×  
/usr/local/bin/python3.10 /Users/di/Desktop/DirtyPython/dirtyPy[2, 4, 6, 8, 0]  
Process finished with exit code 0

```
my_list = '1234567890'
my_list = list(map(int, my_list))
my_list = list(filter(lambda x: x % 2 == 0, my_list))
print(my_list)
```

lection7 ×  
/usr/local/bin/python3.10 /Users/di/Desktop/DirtyPython/dirtyPy[2, 4, 6, 8, 0]  
Process finished with exit code 0

```
my_list = '1234567890'
my_list = [int(i) for i in my_list if int(i) % 2 == 0]
print(my_list)
```

lection7 ×  
/usr/local/bin/python3.10 /Users/di/Desktop/DirtyPython/dirtyPy[2, 4, 6, 8, 0]  
Process finished with exit code 0



# МУДРОСТЬ ОТ СТОУНА

ПРИШЕЛ COMPREHENSION И СКАЗАЛ:  
"ВСЕМ СОСАТЬ!"

# DICT COMPREHENSION

- Точно так же работает и для словарей, только не забывайте использовать фигурные скобки и указывать и ключ и значение
- Если забудете про «ключ : значение», то получите множество
- И только кортеж так не получите – если будете использовать круглые скобки, то получите генератор. А он работает иначе – это отдельная тема.

```
my_list = [i**2 for i in range(10) if i**2 % 2 == 0]
print(my_list)

my_dict = {i: i**2 for i in range(10) if i**2 % 2 == 0}
print(my_dict)

my_set = {i**2 for i in range(10) if i**2 % 2 == 0}
print(my_set)

my_gen = (i**2 for i in range(10) if i**2 % 2 == 0)
print(my_gen)
```

```
lecture7 ×
/usr/local/bin/python3.10 /Users/di/Desktop/DirtyPython/dirtyPython1/Lec-
[0, 4, 16, 36, 64]
{0: 0, 2: 4, 4: 16, 6: 36, 8: 64}
{0, 64, 4, 36, 16}
<generator object <genexpr> at 0x108151e00>
```

# INPLACE



## Изменение переменной на месте

Если мы хотим увеличить переменную, нам не обязательно писать  $a = a + 1$ , можно записать коротко  $a += 1$

Сокращенная запись доступна со всеми операциями

```
59 num = 10
60 print(num)
61 num += 1
62 print(num)
63 num -= 1
64 print(num)
65 num *= 2
66 print(num)
67 num /= 5
68 print(num)
69 num %= 10
70 print(num)
71 num /= 10
72 print(num)
```

```
In: lection7
      ↑ 10
      ↓ 11
      ↵ 10
      ⏴ 20
      ⏵ 4.0
      ⏵ 4.0
      ⏵ 0.4
```

# ДВОЙНОЕ СРАВНЕНИЕ



- В отличие от большинства языков Python позволяет делать множественное сравнение
  - Если мы хотим выполнить какие то действия с а только, если а находится в промежутке между 0 и 100 нам не обязательно использовать оператор and

$$0 < a < 100$$

- Более того, можно даже использовать тройное сравнение

$$0 < a < b < 100$$

- Главное не запутаться!

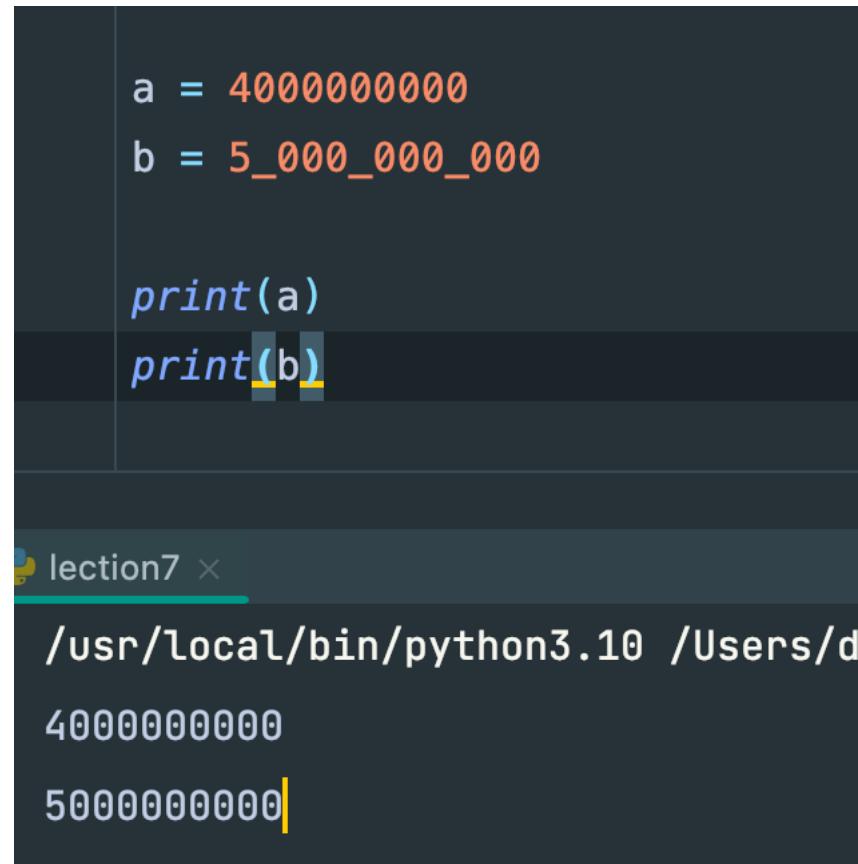
```
a = 40
b = 50

if 0 < a < b < 100:
    print('yep')
```

if 0 < a < b < 100

# РАЗДЕЛЕНИЕ РАЗРЯДНОСТИ

- При работе с большими цифрами, чтобы глазик не запутался можно использовать нижнее подчеркивание для разделения разрядностей – пайтон все поймет



The screenshot shows a code editor window with a dark theme. The code in the editor is:

```
a = 4000000000
b = 5_000_000_000

print(a)
print(b)
```

Below the code, the terminal window shows the execution results:

```
lecture7 ✘
/usr/local/bin/python3.10 /Users/d
4000000000
5000000000
```



# F СТРОКА



- Форматирование строк
- Раньше был метод format, который позволял использовать значение переменных для строк
- Но на смену пришла f-строка, которая очевиднее и прекраснее
- В скобках можно запихнуть не только объект, но и выражение и даже тернарный оператор

```
name = 'STONE'
age = 38
print('Меня зовут {} и мне {} лет'.format(name, age))
```

lecture7 ×  
/usr/local/bin/python3.10 /Users/di/Desktop/DirtyPython/dirtyPython1/Lecture7.py  
Меня зовут STONE и мне 38 лет

```
80 name = 'Ирина'
81 age = 38
82 print(f'Меня зовут {name} и мне {18 if age > 30 else age} лет')
83 age = 25
84 print(f'Меня зовут {name} и мне {18 if age > 30 else age} лет')
```

lecture7 ×  
/usr/local/bin/python3.10 /Users/di/Desktop/DirtyPython/dirtyPython1/Lecture7.py  
Меня зовут Ирина и мне 18 лет  
Меня зовут Ирина и мне 25 лет

# ФОРМАТИРОВАНИЕ ФОРМАТИРОВАНИЯ



- Форматирование указывается после двоеточия (поэтому будьте осторожны с двоеточием, все что идет после него будет считаться форматированием)
- Можно выравнивать < равнение по левому краю, > равнение по правому краю, ^ равнение по центру
- После можно указать количество символов, которое будет обозначать сколько символов выводится под значение
- Если перед знаком форматирования добавить символ, который будет вместо пробела

```
80 name = 'Ирина'
81 age = 38
82 print(f'Меня зовут {name: >20} и мне {18 if age > 30 else age} лет')
83 age = 25
84 print(f'Меня зовут {name: <20} и мне {18 if age > 30 else age} лет')
85 print(f'Меня зовут {name: ^20} и мне {18 if age > 30 else age} лет')
86
```

```
lecture7 ×
↑ /usr/Local/bin/python3.10 /Users/di/Desktop/DirtyPython/dirtyPython1/Lection/lection7
↓ Меня зовут           Ирина и мне 18 лет
↓ Меня зовут Ирина   |   и мне 25 лет
↓ Меня зовут           Ирина и мне 25 лет
```

# ЧТОБЫ СТРОЧКА НЕ РАСПЛОЗАЛАСЬ НА ВСЕЙ ЭКРАН



- Если у нас f-строка просто используем enter – pycharm все сам перенесет, но при выводе оставит одной строкой
- Или мы можем использовать служебный символ /n - тогда и при выводе строки будут разными
- Без f строки можно использовать конкатенацию + или /
- И в print про end не забываем!
- В скобках можно просто переносить – даже без /n

The screenshot shows two code snippets in PyCharm and their corresponding terminal outputs.

**Top Snippet:**

```
80     name = 'Ирина'
81     age = 38
82     print(f'Меня зовут {name} и мне {age if age > 30 else age} лет'
83           f'dfgdfgf sdfgdfgfd')
84
85     print(f'Меня зовут {name} и мне {age if age > 30 else age} лет\n'
86           f'dfgdfgf sdfgdfgfd')
87
```

**Terminal Output:**

```
lecture7 ×
/usr/local/bin/python3.10 /Users/di/Desktop/DirtyPython/dirtyPython1/Lecture
↓
Меня зовут Ирина           и мне 18 летdfgdfgf sdfgdfgfd
Меня зовут Ирина           и мне 18 лет
dfgdfgf sdfgdfgfd
```

**Bottom Snippet:**

```
88     print('12312 123243 1234534535 12343455 123434546'
89           + 'asfdg asdfg fdgghj hgj')
90
```

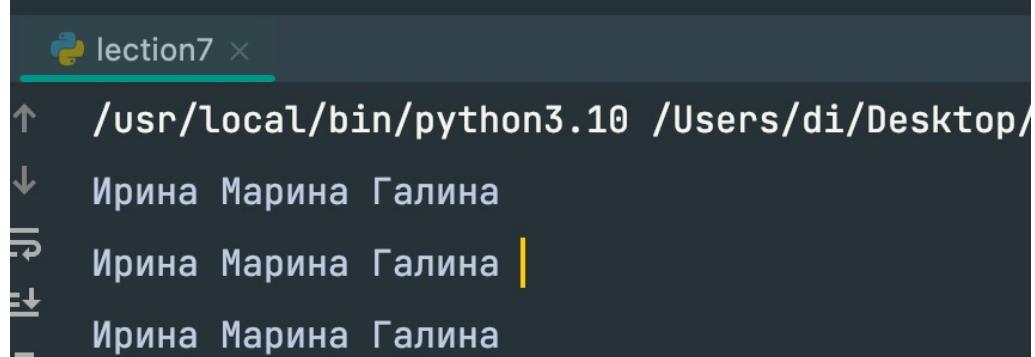
**Terminal Output:**

```
lecture7 ×
/usr/local/bin/python3.10 /Users/di/Desktop/DirtyPython/dirtyPython1/Lecture/lection7.py
12312 123243 1234534535 12343455 123434546asfdg asdfg fdgghj hgj
Process finished with exit code 0
```

# ПЕЧАТАЕМ СПИСОК КРАСИВО

- И опять же есть несколько способов:
- ✓ перечисляем все элементы через запятую
- ✓ проходимся циклом for
- ✓ используем распаковку
- И не забываем про sep и end, чтобы отрегулировать разделение и вывод элементов

```
90
91     name = ['Ирина', 'Марина', 'Галина']
92
93     print(name[0], name[1], name[2])
94
95     for n in name:
96         print(n, end=' ')
97     print()
98     print(*name)
99
```



# КОММЕНТАРИИ И ДОКУМЕНТАЦИЯ



- Комментарии пишем через # там, где нам удобно
- Описание функции – после ее объявления ставим тройные кавычки и все описываем
- Затем мы можем посмотреть описание вызвав `help(имя_функции)`
- Встроенные методы кстати тоже можно посмотреть при помощи `help`, у них-то с документацией все в порядке

```
def pow(a: int, b: int) -> int:  
    """  
    эта функция возводит число a в степень b с использованием рекурсии  
    :param a: int  
    :param b: int  
    :return: int  
    """  
    return pow(a, b-1)*a if b != 1 else a  
  
help(pow)
```

lection7 x

```
pow(a: int, b: int) -> int  
эта функция возводит число a в степень b с использованием рекурсии  
:param a: int  
:param b: int  
:return: int
```

# УСЛОВИЯ ДЛЯ IF И WHILE



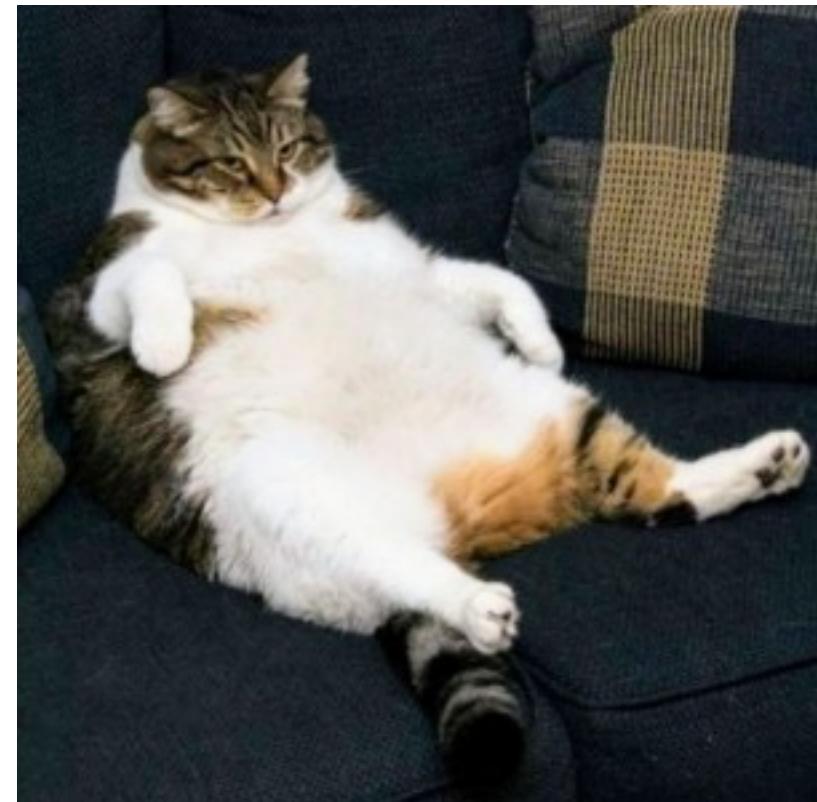
- Нам не обязательно ставить логическое выражение, мы можем вполне успешно использовать и переменную функцию, и результат вызова функции, потому как в Python все так или иначе сводиться True/False

```
2
3 a = 88
4
5 if not a%2:
6     print('Четное')
```

# ЛЕННЫЙ IF

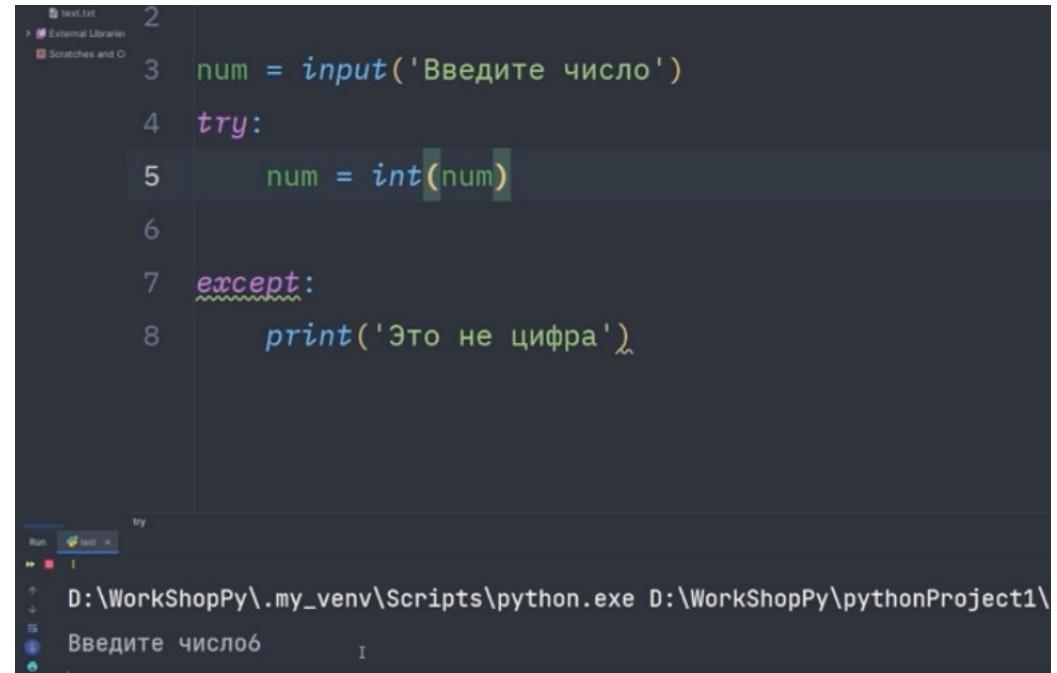


- В логических выражениях с and if проверяет сначала первое условие – если оно False, то дальше код даже не отрабатывается
- В логических с выражениях с or if проверяет сначала первое условие – если оно True, то дальше не проверяется.



# TRY EXCEPT

- Конструкция для отлова исключений, которая поможет вам не только с этим (пусть и говорят, что это дурной тон)
- В try мы закладываем действие, которое должна произвести программа.
- Если выполнение этой части кода приводит к ошибке, то программа не ложиться, а переходит к except – где мы например можем вывести сообщение об ошибке либо провернуть альтернативный код с полученными данными



```
2 num = input('Введите число')
3
4 try:
5     num = int(num)
6
7 except:
8     print('Это не цифра')
```

The screenshot shows a Python script named 'test' in a code editor. The script contains a try-except block. It prompts the user for input, tries to convert it to an integer, and prints an error message if it fails. The code editor interface includes tabs for 'Run' and 'External Libraries'. Below the editor, a terminal window shows the command 'python test' being run, followed by the prompt 'Введите число' (Enter number) and the user's input 'I'.



# МИНИМАЛЬНОЕ И МАКСИМАЛЬНОЕ ЗНАЧЕНИЕ INT



- Нам поможет float и inf
- Заполучив минус и плюс бесконечность вы можете делать с ними, что хотите (точнее то, что вам позволит математика – на ноль все еще делить не рекомендуется)

```
2
3
4 max_value = float('inf')
5 min_value = float('-inf')
6
7 print(min_value > 0)
8 print(max_value)
```

The screenshot shows a Python code editor with a script named 'test.txt'. The code defines two variables: 'max\_value' set to positive infinity ('inf') and 'min\_value' set to negative infinity ('-inf'). It then prints the value of 'min\_value' greater than zero and the value of 'max\_value'. The output window shows the results of the execution.

```
Run test.py
D:\WorkShopPy\.my_venv\Scripts\python.exe D:\WorkShopPy\pythonProject1\tes
False
inf
```



# МУДРОСТЬ ОТ СТОУНА

"НЕ БОЙТЕСЬ МАТЕМАТИКИ, БОЙТЕСЬ ЛОГИКИ!  
ЛОГИКА СТРАШНЕЕ: В МАТЕМАТИКЕ ЕСТЬ  
ФОРМУЛЫ, В ЛОГИКЕ НЕТ НИ\*УЯ"

# ИЧТО ДАЛЬШЕ?



D1RTY  
РУТНД

ПЕЙТЕ ЧАЙ

meme-arsenal.ru

# И ЧТО ДАЛЬШЕ?

- ДЕЛАЕМ ДОМАШКУ, ОНА КЛАССНАЯ
- НАС ДАЛЬШЕ ЖДУТ ПРЯМ ОЧЕНЬ ИНТЕРЕСНЫЕ ЛЕКЦИИ - ГОТОВЬТЕ МОЗГИ )



D1RTY  
РУТНОН