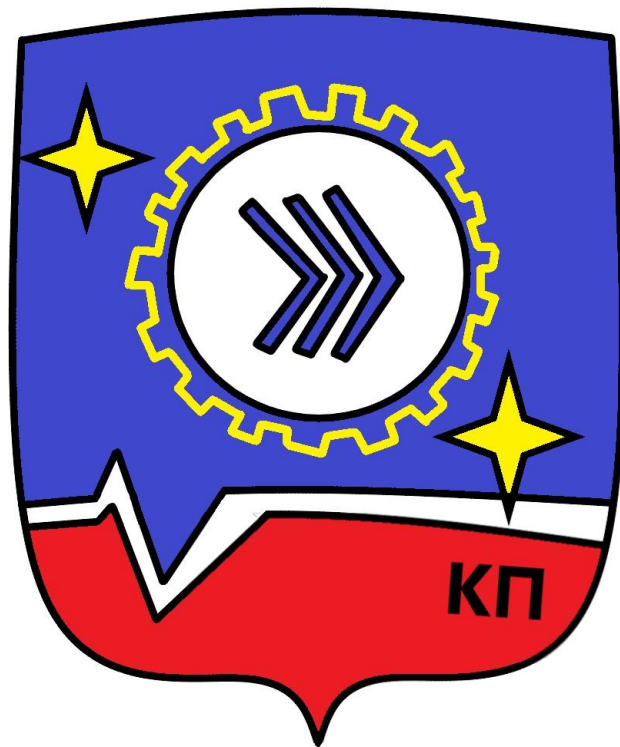


КЛУБ ПРОГРАММИСТОВ Г.КОРОЛЁВА



КУРС “Программирование на PYTHON”



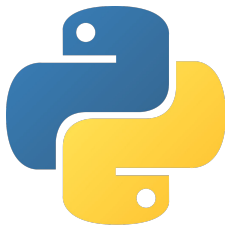


РЕГЛАМЕНТ



1. Вопросы можно задавать
в любое время





РЕГЛАМЕНТ



1. Вопросы можно задавать в любое время

**2. Встречаемся 1 раз в неделю
(или реже) по субботам**





РЕГЛАМЕНТ



1. Вопросы можно задавать в любое время
2. Встречаемся 1 раз в неделю (или реже) по субботам

**3. Две лекции по 45 минут
+ дополнительная лекция**






РЕГЛАМЕНТ



1. Вопросы можно задавать в любое время
2. Встречаемся 1 раз в неделю (или реже) по субботам
3. Две лекции по 45 минут + дополнительная лекция

**4. Домашние задания – ДА,
аттестация – ДА,
сертификаты – ДА**


**KEEP
CALM
AND
DO YOUR
HOMEWORK**



РЕГЛАМЕНТ



1. Вопросы можно задавать в любое время
2. Встречаемся 1 раз в неделю (или реже) по субботам
3. Две лекции по 45 минут + дополнительная лекция
4. Домашние задания - ДА, аттестация - ДА, сертификаты - ДА

5. Площадка для общения



https://vk.com/python_korolev



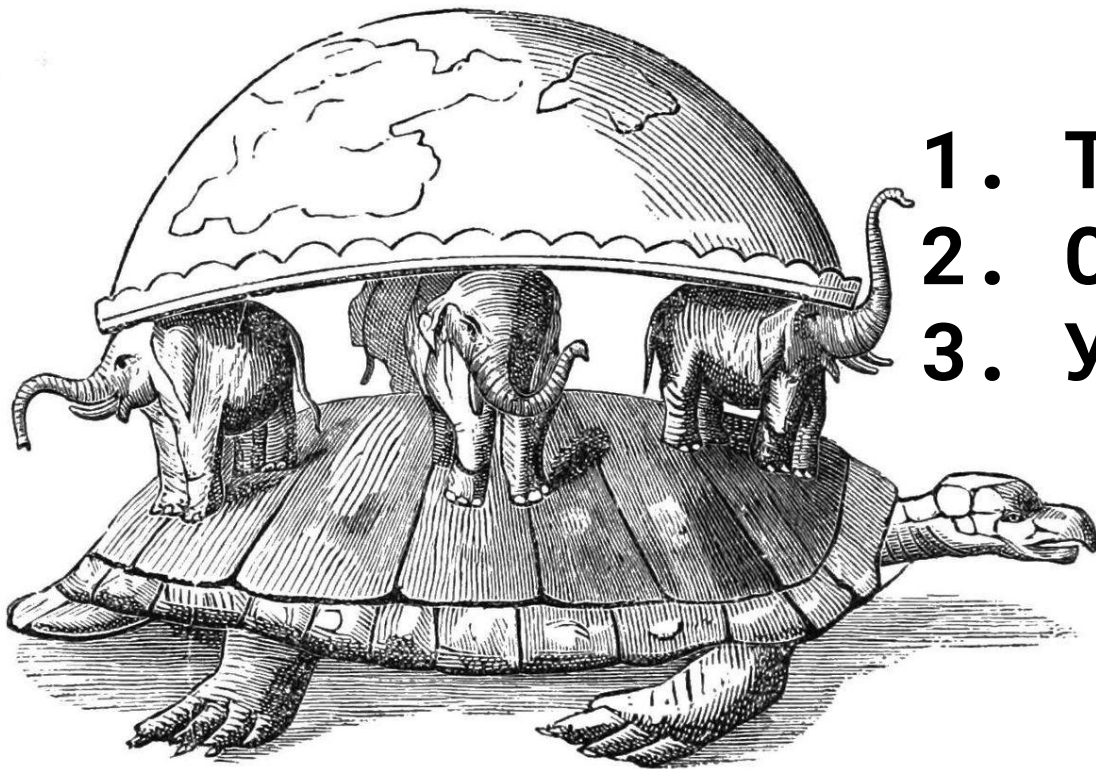
+7 (909) 820-56-18



ЧТО ТАКОЕ ЯП



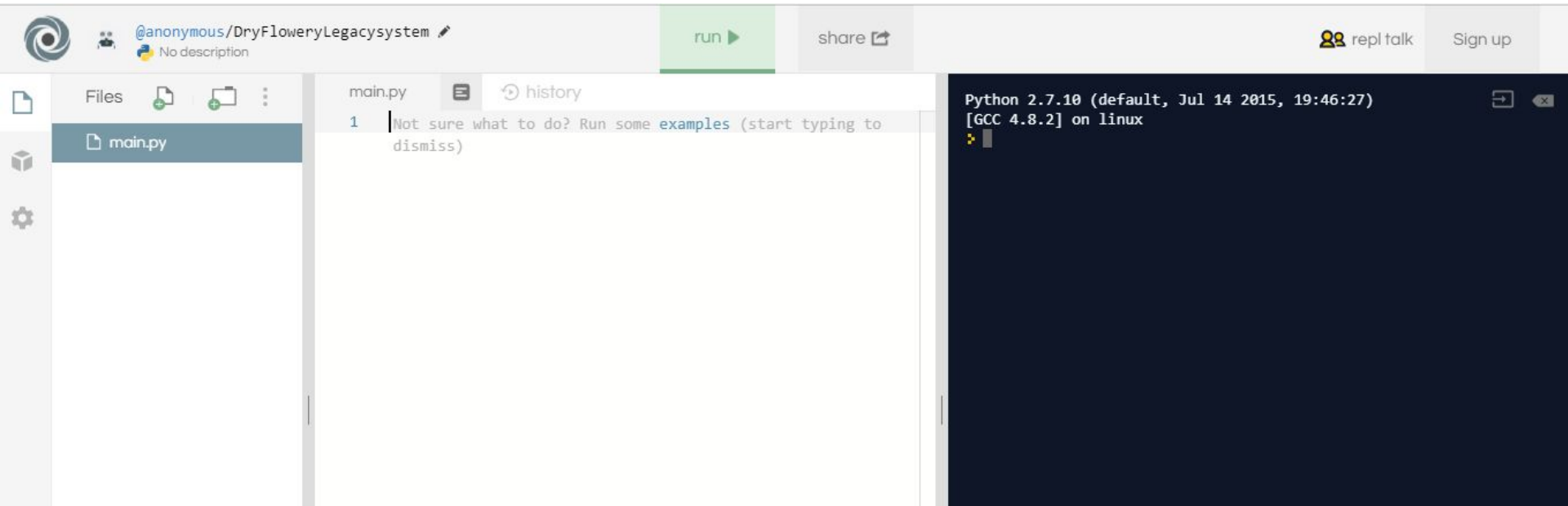
1. Типы данных
2. Структуры
3. Управляющие конструкции





PYTHON ONLINE

repl.it





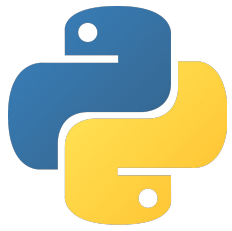
PYTHON и command line



```
C:\> Command Prompt - python

Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Users\Eric>python
Python 2.7.12 (v2.7.12:d33e0cf91556, Jun 27 2016, 15:19:22) [MSC v.1500 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> pytest.py
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'pytest' is not defined
>>> _
```



ПАРАДИГМЫ ПРОГРАММИРОВАНИЯ



Императивное программирование

Структурное программирование

Функциональное программирование

Объектно-ориентированное

программирование



ТИПЫ ДАННЫХ



int

ЦЕЛОЕ - INTEGER

1

2

3



ПРОЦЕДУРНОЕ ПРОГРАММИРОВАНИЕ



Выполнение программы сводится к **последовательному выполнению** операторов с целью преобразования **исходного состояния** памяти, то есть значений исходных данных, в заключительное, то есть **в результаты**.

Таким образом, с точки зрения программиста имеются программа и память, причем первая последовательно обновляет содержимое последней.



.py



hello.py

```
print('Hello World!')
```

```
c:\python.exe hello.py
```

```
print('Hello', 'World!')
```



ТИПЫ ДАННЫХ



str

СТРОКА - STRING

"Hello"

'Hello'

'''hello'''

"""hello"""

r"raw_string"



ТИПЫ ДАННЫХ неизменяемость



int
str

a = 5

b = 'MAN'

b[1]



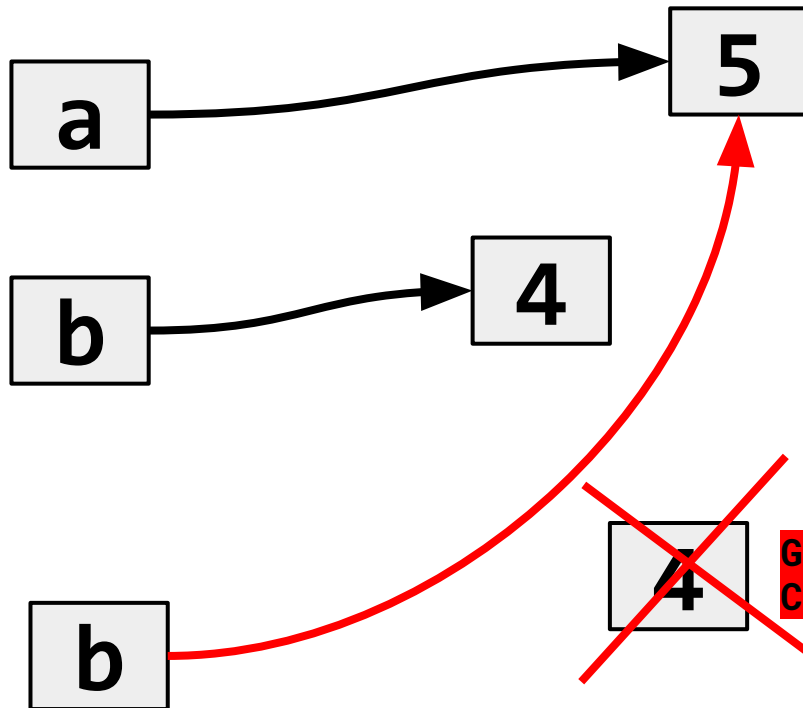
ТИПЫ ДАННЫХ

присваивание значений

a = 5

b = 4

b = a



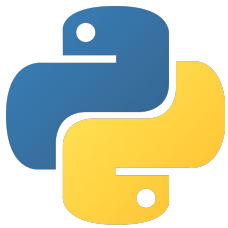
id()



СТРУКТУРЫ СПИСОК LIST



```
x = [1, 2, "a", a, [1, 3]]  
y = list(1, 2, "g")  
len(x)  
x[3]  
x[1] = 4
```



СТРУКТУРЫ КОРТЕЖ TUPLE

```
z = 1,  
x = (1, 2, "a", a, [1, 3])  
y = tuple(1, 2, "g")  
len(x)  
x[3]  
x[1] = 4
```



СТРУКТУРЫ LIST МЕТОДЫ

```
a = [1, 2, 3]
a.append(4) # [1, 2, 3, 4]
list.append(a, 4)
a.insert(1, 'yeah')
a.remove('yeah')
del a[0]
```



ЛОГИЧЕСКИЕ ОПЕРАЦИИ



True - ПРАВДА - 1
False - ЛОЖЬ - 0

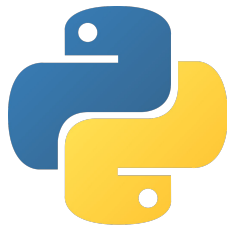


ЛОГИЧЕСКИЕ ОПЕРАЦИИ

a is b - идентичность
(сравнение адресов в памяти)

is / is not
None

a == b
(сравнение значений)



ЛОГИЧЕСКИЕ ОПЕРАЦИИ



a = 4

b = 10

a == b # False

a > b # False

a >= b # False

a < b # True

a <= b # True

a != b # True

a <= 5 <= b



ЛОГИЧЕСКИЕ ОПЕРАЦИИ

принадлежность группе

`in` / `not in`



```
a = [1, 6, 7, 'j', 'alpha']  
7 in a # True
```

```
b = 'Я хочу спать!'  
'ь' not in b # False
```




ЛОГИЧЕСКИЕ ОПЕРАЦИИ

операторы

and, or, not



`1 < 2 and 2 < 4`
`3 > 4 or 2 < 5`
`not 3 < 2`



УПРАВЛЕНИЕ ПОТОКОМ ВЕТВЛЕНИЕ



```
if условие:  
    выражение  
elif условие:  
    выражение  
elif условие:  
    выражение  
else:  
    выражение
```



УПРАВЛЕНИЕ ПОТОКОМ ЦИКЛ WHILE



while условие: # True / False
выражение

while True:

break
continue



УПРАВЛЕНИЕ ПОТОКОМ ЦИКЛ FOR .. IN



**for переменная in итератор:
выражение**

**break
continue**



УПРАВЛЕНИЕ ПОТОКОМ ОБРАБОТКА ИСКЛЮЧЕНИЙ



try:

выражение

except:

выражение

except TypeError:

except ValueError as err:



УПРАВЛЕНИЕ ПОТОКОМ АРИФМЕТИКА



5 + 6

11

a = 1

3 - 7

-4

a += 2 # 3

4 * 8

32

a = a + 2

3 / 2

1.5

5 // 2

2

5 % 2

1

3 ** 2

9



“АРИФМЕТИКА”

“Hello” + “World” = “HelloWorld”

“Rx” * 3 = “RxRxRx”

[1, 2, 3] + [2] = [1, 2, 3, 2]

[1, 2] * 2 = [1, 2, 1, 2]



ВВОД ДАННЫХ

```
x = input( 'Введите число: ' )  
int(x)
```




ФУНКЦИИ



```
def имя(аргументы):  
    операции  
    return результат
```



import



```
import random  
print(random.randint(1, 6))  
print(random.choice("любит", "не любит"))
```

```
from random import randint  
print(randint(0, 41))
```



`range()`



`range(10)`

`range(1, 10)`

`range(1, 10, 3)`

`range(100, 10, -2)`



ЕЩЕ СТРУКТУРЫ МНОЖЕСТВА

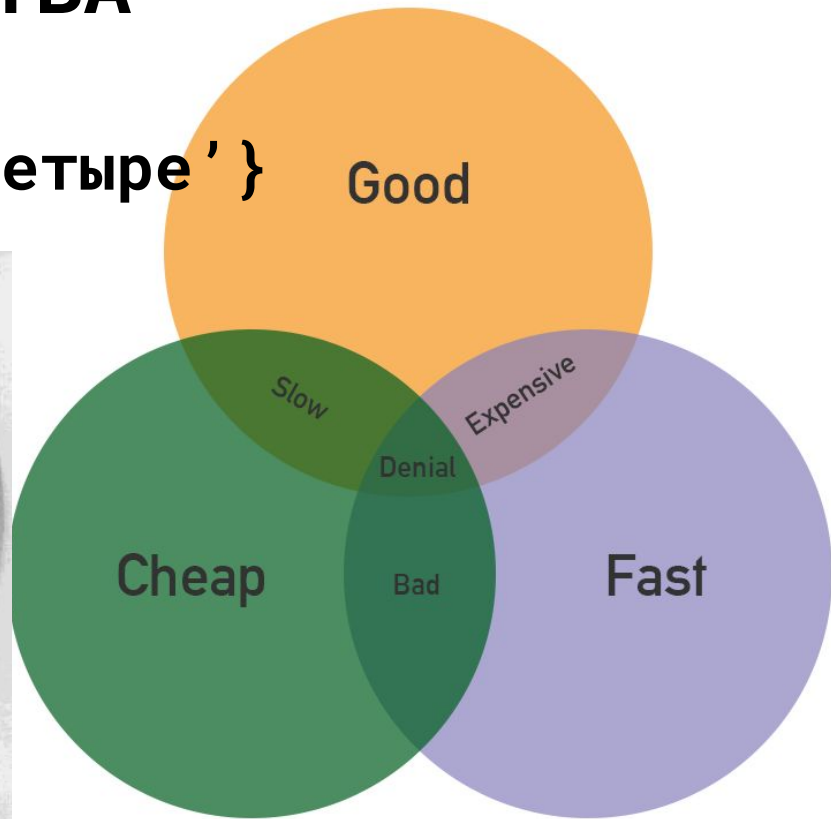


set()

```
some_set = {1, 2, 3, 'четыре'}
```



Эйлер
Венн





ЕЩЕ СТРУКТУРЫ СЛОВАРИ



dict()

```
some_dict = {  
    1: 2,  
    2: 'собака',  
    3: [1, 2, 'кошка'],  
    'четыре': None}
```

Нет сортировки
Быстрый поиск

```
some_dict[3]  
some_dict['четыре']
```

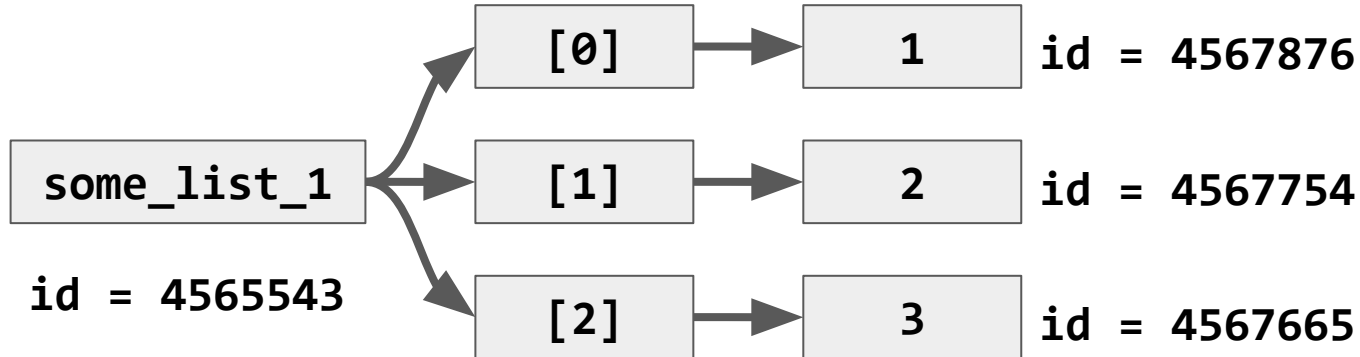


КОПИРОВАНИЕ СПИСКОВ



```
some_list_1 = [1, 2, 3]
```

```
id = id(some_list_1[0])
```

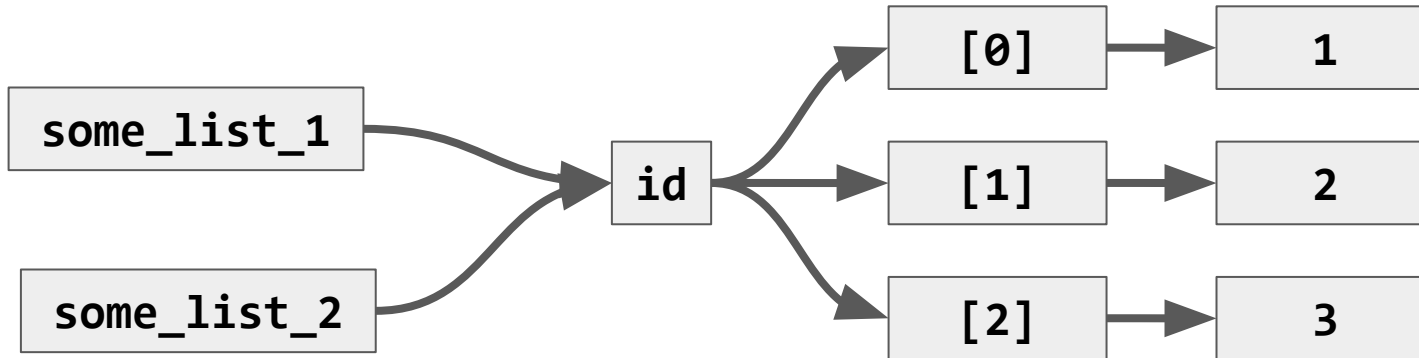




КОПИРОВАНИЕ СПИСКОВ



```
some_list_1 = [1, 2, 3]  
some_list_2 = some_list_1
```

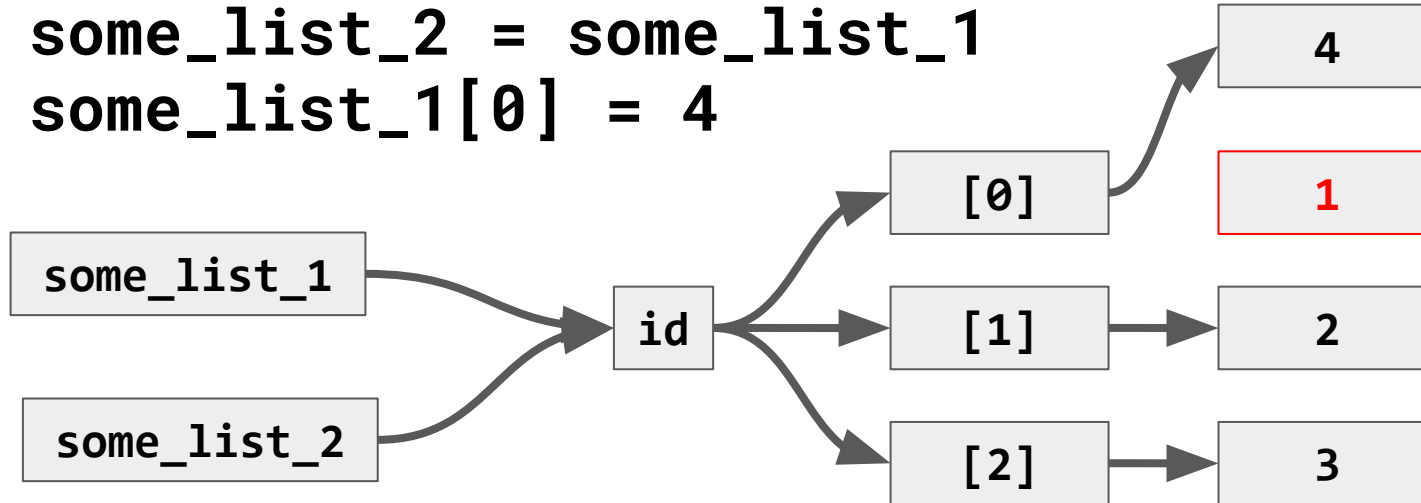




КОПИРОВАНИЕ СПИСКОВ



```
some_list_1 = [1, 2, 3]  
some_list_2 = some_list_1  
some_list_1[0] = 4
```



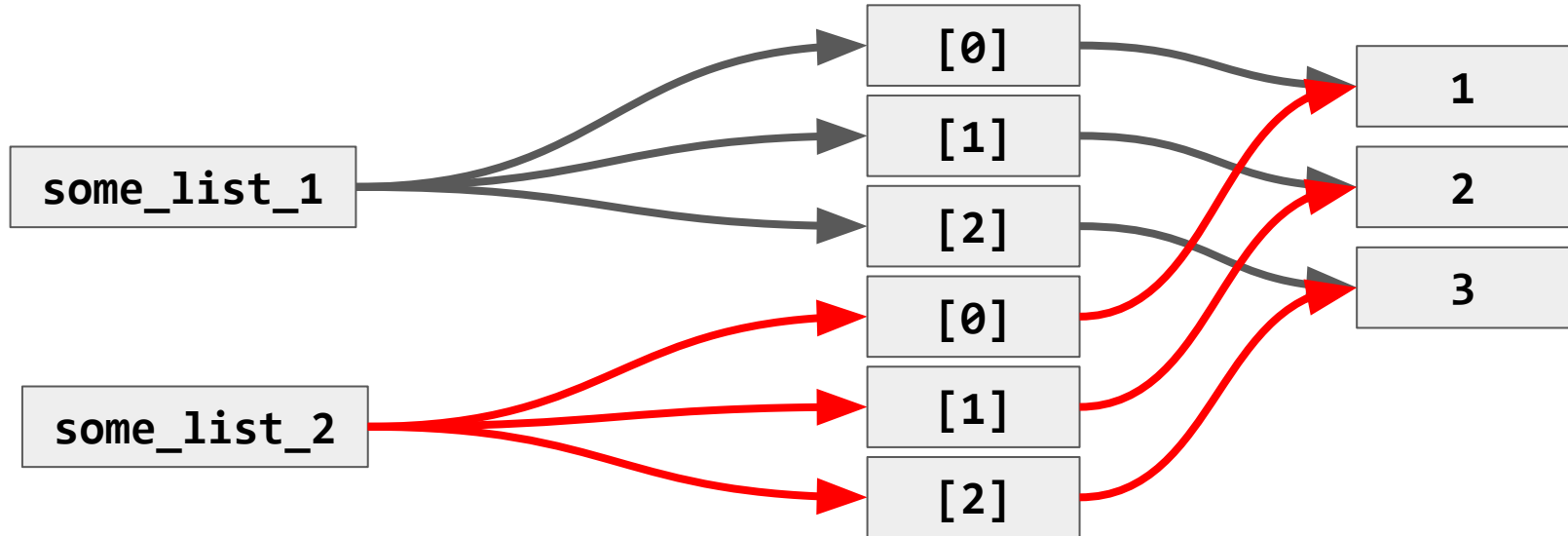
```
some_list_2[0] : 4
```




КОПИРОВАНИЕ СПИСКОВ



```
some_list_1 = [1, 2, 3]  
some_list_2 = some_list_1[:]
```

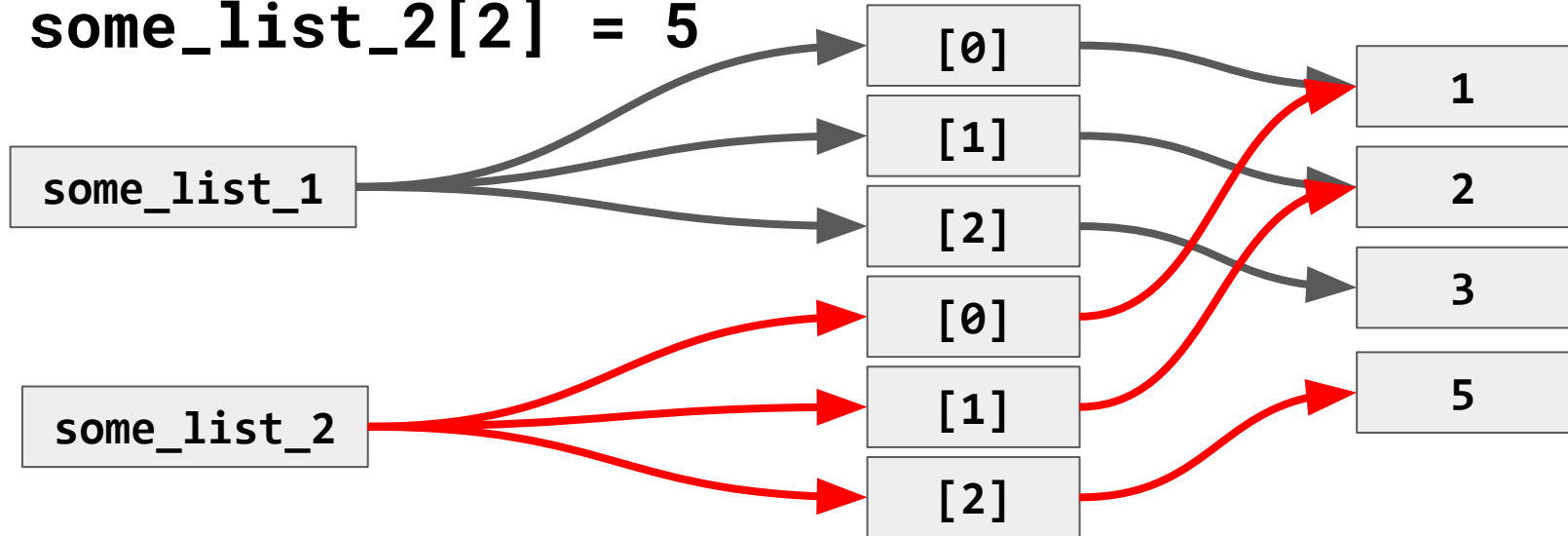




КОПИРОВАНИЕ СПИСКОВ



```
some_list_1 = [1, 2, 3]  
some_list_2 = some_list_1[:]  
some_list_2[2] = 5
```

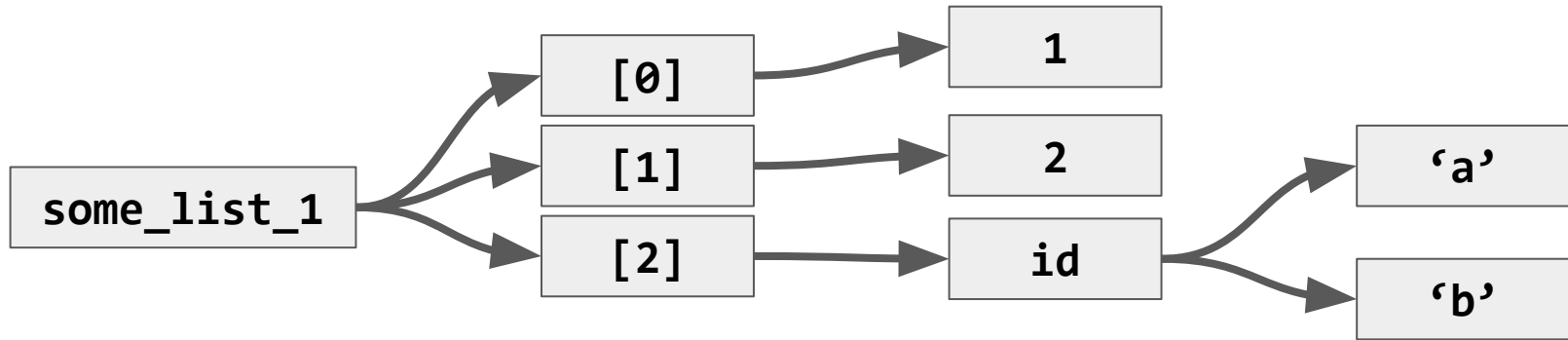




КОПИРОВАНИЕ СПИСКОВ



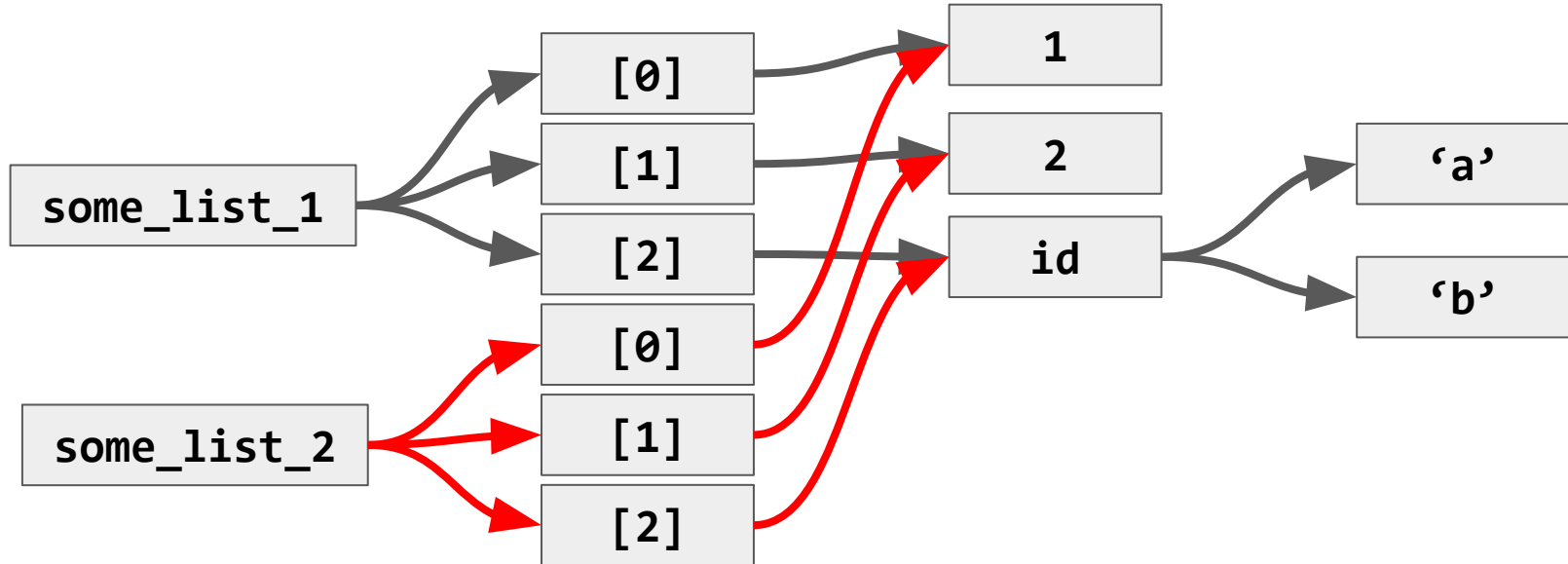
```
some_list_1 = [1, 2, ['a', 'b']]
```





КОПИРОВАНИЕ СПИСКОВ

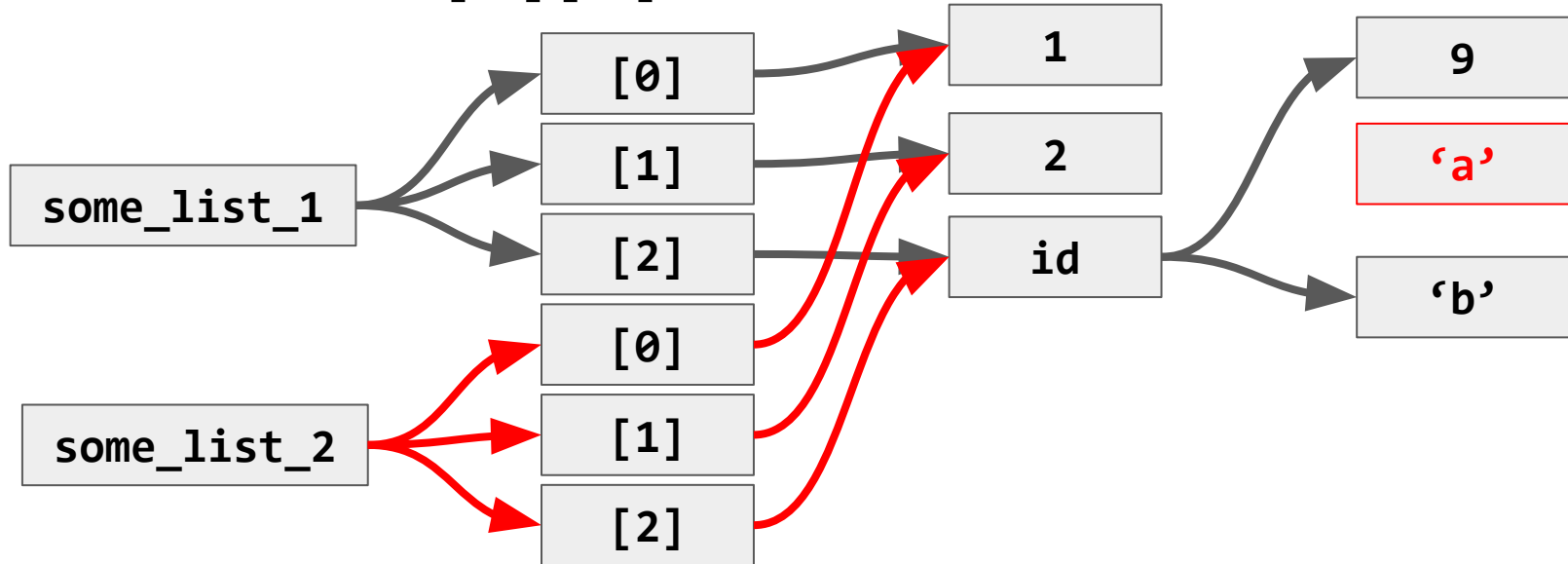
```
some_list_1 = [1, 2, ['a', 'b']]  
some_list_2 = some_list_1
```





КОПИРОВАНИЕ СПИСКОВ

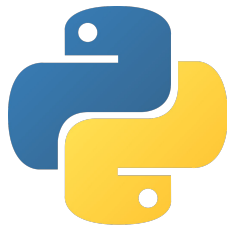
```
some_list_1 = [1, 2, ['a', 'b']]  
some_list_2 = some_list_1  
some_list_1[2][0] = 9
```





ПРИСТУПИМ К ПРОГРАММИРОВАНИЮ





ПРИСТУПИМ К ПРОГРАММИРОВАНИЮ



1. Парсим файл с текстом
2. Решаем задачки



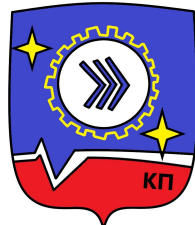
ПАРСЕР



1. Открыть файл
2. Прочитать содержимое
3. Обработать по определенным правилам



ПАРСЕР



ТОМ ПЕРВЫЙ

ЧАСТЬ ПЕРВАЯ

I.

- Eh bien, mon prince. Gknes et Lucques ne sont plus que des apanages, des поместья, de la famille Buonaparte. Non, je vous prйviens, que si vous ne me dites pas, que nous avons la guerre, si vous vous permettez encore de pallier toutes les infamies, toutes les atrocitйs de cet Antichrist (ma parole, j'y crois) - je ne vous connais plus, vous n'ktes plus mon ami, vous n'ktes plus мой верный раб, comme vous dites. 1 (См. сноски в конце части) Ну, здравствуйте, здравствуйте. Je vois que je vous fais peur, 2 садитесь и рассказывайте.

Так говорила в июле 1805 года известная Анна Павловна Шерер, фрейлина и приближенная императрицы Марии Феодоровны, встречая важного и чиновного князя Василия, первого приехавшего на ее вечер. Анна Павловна кашляла несколько дней, у нее был грипп, как она говорила (грипп был тогда новое слово, употреблявшееся только редкими). В записочках, разосланных утром с красным лакеем, было написано без различия во всех:

"Si vous n'avez rien de mieux a faire, M. le comte (или mon prince), et



ПАРСЕР



```
f = open('file.txt', 'r', encoding='utf8')
```

Менеджер контекста – гарантия освобождения ресурса

```
with open() as f:  
    pass
```



ПАРСЕР



```
with open('lt1.txt') as wp_file:  
    print(wp_file)
```

```
<_io.TextIOWrapper  
name='lt1.txt'  
mode='r'  
encoding='cp1251'>
```



ПАРСЕР



```
with open('lt1.txt') as wp_file:  
    print(wp_file.read())
```



ПАРСЕР



```
with open('lt1.txt') as wp_file:  
    wp_text = wp_file.read()  
    wp_list = wp_text.split('.')  
    print(len(wp_list))
```



ПАРСЕР



```
with open('lt1.txt') as wp_file:  
    wp_text = wp_file.read()  
    wp_list = wp_text.split('.')  
    for sentence in wp_list:  
        print(sentence)
```



ПАРСЕР



```
with open('lt1.txt') as wp_file:  
    wp_text = wp_file.read()  
    wp_list = wp_text.split('.')  
    for index, sentence in enumerate(wp_list):  
        print(index, sentence)
```

Распаковывание

```
a, b, c = [1, 2, 3]
```

```
a, b, *c = [1, 2, 3, 4, 5]
```

```
a, *b, c = [1, 2, 3, 4, 5]
```



ПАРСЕР



```
with open('lt1.txt') as wp_file:
    wp_text = wp_file.read()
    wp_list = wp_text.split('.')
    for index, sentence in enumerate(wp_list):
        print(    index,
                  sentence.replace('\n', ' '))
```




ПАРСЕР



```
with open('lt1.txt') as wp_file:  
    wp_text = wp_file.read()  
    wp_list = wp_text.split('.')  
    while True:  
        search_word = input('Введите слово для поиска: ')  
        search_word = search_word.strip()
```



ПАРСЕР



```
with open('lt1.txt') as wp_file:
    wp_text = wp_file.read()
    wp_list = wp_text.split('.')
    while True:
        search_word = input('Введите слово для поиска: ')
        search_word = search_word.strip()
        for index, sentence in enumerate(wp_list):
            if search_word in sentence:
                print(index,
                    sentence.replace('\n', ' ').strip())
```



ПАРСЕР



```
with open('lt1.txt') as wp_file:
    wp_text = wp_file.read()
    wp_list = wp_text.split('.')
    while True:
        search_word = input('Введите слово для поиска: ')
        search_word = search_word.strip()
        for index, sentence in enumerate(wp_list):
            if search_word in sentence:
                print(index,
                    sentence.replace('\n', ' ').strip())
                break
```



СЕРТИФИКАЦИЯ ВОПРОС



Язык **Python**:

Вариант 1 поощряет повторное использование кода

Вариант 2 слабо типизирован

Вариант 3 сильно типизирован

Вариант 4 является высокоуровневым языком программирования

Вариант 5 является низкоуровневым языком программирования



СЕРТИФИКАЦИЯ ВОПРОС



Язык **Python**:

Вариант 1 поощряет повторное использование кода

Вариант 2 слабо типизирован

Вариант 3 сильно типизирован

Вариант 4 является высокоуровневым языком программирования

Вариант 5 является низкоуровневым языком программирования



СЕРТИФИКАЦИЯ ВОПРОС



Язык **Python** поддерживает следующие парадигмы программирования:

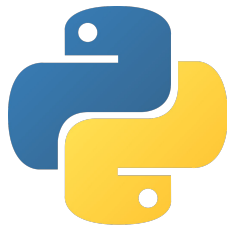
Вариант 1 модульное программирование

Вариант 2 императивное программирование

Вариант 3 функциональное программирование

Вариант 4 объектно-ориентированное программирование

Вариант 5 логическое программирование



СЕРТИФИКАЦИЯ ВОПРОС



Язык **Python** поддерживает следующие парадигмы программирования:

Вариант 1 модульное программирование

Вариант 2 императивное программирование

Вариант 3 функциональное программирование

Вариант 4 объектно-ориентированное программирование

Вариант 5 логическое программирование



СЕРТИФИКАЦИЯ ВОПРОС



Из приведенных ниже высказываний укажите истинное:

Вариант 1 перед использованием переменной она должна быть инициализирована каким-либо значением

Вариант 2 перед использованием переменной она должна быть определена с указанием типа и инициализирована каким-либо значением

Вариант 3 перед использованием переменной она должна быть определена с указанием типа

Вариант 4 перед использованием переменной она будет автоматически проинициализирована значением по умолчанию

Вариант 5 переменные не обязаны быть инициализированы каким-либо значением перед использованием, а тип переменной определяется в зависимости от контекста



СЕРТИФИКАЦИЯ ВОПРОС



Из приведенных ниже высказываний укажите истинное:

Вариант 1 перед использованием переменной она должна быть инициализирована каким-либо значением

Вариант 2 перед использованием переменной она должна быть определена с указанием типа и инициализирована каким-либо значением

Вариант 3 перед использованием переменной она должна быть определена с указанием типа

Вариант 4 перед использованием переменной она будет автоматически проинициализирована значением по умолчанию

Вариант 5 переменные не обязаны быть инициализированы каким-либо значением перед использованием, а тип переменной определяется в зависимости от контекста



ИДЕНТИФИКАТОРЫ



a = 5

- **a** – идентификатор, имя
- **=** – “присваивание”: связывание ссылки на объект с объектом в памяти, который хранит данные.



ИДЕНТИФИКАТОРЫ ТРЕБОВАНИЯ К ИМЕНИ



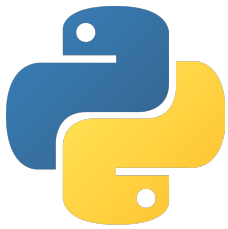
- **произвольная длина**
- начинается с алфавитного символа или с подчеркивания (PEP 3131)
- остальные символы в имени – алфавитные, цифровые, _
- name и Name – разные имена
- имя не должно совпадать с ключевыми словами python and, as, def, False, if, not, or, try, pass, ...
- не использовать предопределенные имена – int, str, math, name – проверка через dir(__builtins__)
- не нужно начинать имя с символа подчеркивания
- _ – результат последнего вычисления либо пустое имя



ИДЕНТИФИКАТОРЫ ТРЕБОВАНИЯ К ИМЕНИ



- произвольная длина
- **начинается с алфавитного символа
или с подчеркивания (PEP 3131)**
- остальные символы в имени – алфавитные, цифровые, _
- name и Name – разные имена
- имя не должно совпадать с ключевыми словами python
and, as, def, False, if, not, or, try, pass, ...
- не использовать predefined имена – int, str,
math, name – проверка через dir(__builtins__)
- не нужно начинать имя с символа подчеркивания
- _ – результат последнего вычисления либо пустое имя



ИДЕНТИФИКАТОРЫ ТРЕБОВАНИЯ К ИМЕНИ



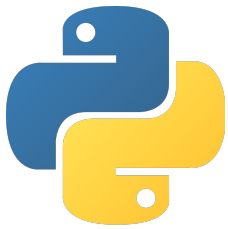
- произвольная длина
- начинается с алфавитного символа или с подчеркивания (PEP 3131)
- **остальные символы в имени – алфавитные, цифровые, _**
- name и Name – разные имена
- имя не должно совпадать с ключевыми словами python and, as, def, False, if, not, or, try, pass, ...
- не использовать predefined имена – int, str, math, name – проверка через dir(__builtins__)
- не нужно начинать имя с символа подчеркивания
- _ – результат последнего вычисления либо пустое имя



ИДЕНТИФИКАТОРЫ ТРЕБОВАНИЯ К ИМЕНИ



- произвольная длина
- начинается с алфавитного символа или с подчеркивания (PEP 3131)
- остальные символы в имени – алфавитные, цифровые, _
- **name и Name – разные имена**
- имя не должно совпадать с ключевыми словами python and, as, def, False, if, not, or, try, pass, ...
- не использовать predefined имена – int, str, math, name – проверка через dir(__builtins__)
- не нужно начинать имя с символа подчеркивания
- _ – результат последнего вычисления либо пустое имя



ИДЕНТИФИКАТОРЫ ТРЕБОВАНИЯ К ИМЕНИ



- произвольная длина
- начинается с алфавитного символа или с подчеркивания (PEP 3131)
- остальные символы в имени – алфавитные, цифровые, _
- name и Name – разные имена
- **имя не должно совпадать с ключевыми словами python and, as, def, False, if, not, or, try, pass, ...**
- не использовать predefined имена – int, str, math, name – проверка через dir(__builtins__)
- не нужно начинать имя с символа подчеркивания
- _ – результат последнего вычисления либо пустое имя



ИДЕНТИФИКАТОРЫ ТРЕБОВАНИЯ К ИМЕНИ



- произвольная длина
- начинается с алфавитного символа или с подчеркивания (PEP 3131)
- остальные символы в имени – алфавитные, цифровые, _
- name и Name – разные имена
- имя не должно совпадать с ключевыми словами python and, as, def, False, if, not, or, try, pass, ...
- не использовать predefined имена – int, str, math, name – проверка через dir(__builtins__)
- не нужно начинать имя с символа подчеркивания
- _ – результат последнего вычисления либо пустое имя



ИДЕНТИФИКАТОРЫ ТРЕБОВАНИЯ К ИМЕНИ



- произвольная длина
- начинается с алфавитного символа или с подчеркивания (PEP 3131)
- остальные символы в имени – алфавитные, цифровые, _
- name и Name – разные имена
- имя не должно совпадать с ключевыми словами python and, as, def, False, if, not, or, try, pass, ...
- не использовать predefined имена – int, str, math, name – проверка через dir(__builtins__)
- **не нужно начинать имя с символа подчеркивания**
- _ – результат последнего вычисления либо пустое имя



ИДЕНТИФИКАТОРЫ ТРЕБОВАНИЯ К ИМЕНИ



- произвольная длина
- начинается с алфавитного символа или с подчеркивания (PEP 3131)
- остальные символы в имени – алфавитные, цифровые, _
- name и Name – разные имена
- имя не должно совпадать с ключевыми словами python and, as, def, False, if, not, or, try, pass, ...
- не использовать предопределенные имена – int, str, math, name – проверка через dir(__builtins__)
- не нужно начинать имя с символа подчеркивания
- **_ – результат последнего вычисления либо пустое имя**



Целочисленные типы



int 0, 1

0 - False

bool True, False

Неизменяемые

True + True = 2



int



- размер ограничен только объемом памяти
- Системы счисления:
 - двоичная 0b00011101
 - восьмеричная 0o54372310
 - шестнадцатеричная 0xDEF45



int ОПЕРАЦИИ



abs(int) – абсолютное значение

+ **+=**

- **-=**

a = a + 1

/ **/=**

a += 1

// **//=**

% **%=**

a = int(7.6)

****** ****=**

a = int('7')



`int` ФУНКЦИИ



`abs(int)` – абсолютное значение
`divmod(x, y)` – частное и остаток
`pow(x, y)` – **
`round(x, [n])` – округление до n

`x = int()` # `x = 0`



`int` ПРЕОБРАЗОВАНИЯ



`bin(int)` – двоичное представление
`hex(int)` – 16-ное представление
`oct(int)` – 8-ное представление
`int(x)`
`int('0001010', 2)`



`int`

БИТОВЫЕ ОПЕРАЦИИ



`a | b` – OR (ИЛИ)

`a ^ b` – XOR (исключающая ИЛИ)

`a & b` – AND (И)

`a << b` – сдвиг влево на `b`

`a >> b` – сдвиг вправо на `b`

`-a` – инверсия



`int` БИТОВЫЕ ОПЕРАЦИИ

a	b	OR	XOR	AND
0	0	0	0	0
0	1	1	1	0
1	0	1	1	0
1	1	1	0	1



`int` БИТОВЫЕ ОПЕРАЦИИ

`0b000010 << 1 = 0b000100`

`2 << 1 = 4`

`0b1100 >> 1 = 0b0110`

`12 >> 1 = 6`

`0b1100 >> 2 = 0b0011`

`12 >> 2 = 3`



int ЛОГИЧЕСКИЕ ОПЕРАЦИИ



True

False

a = bool('some')

Ленивые операции

exp1 and exp2 - если exp1 = 0

exp1 or exp2 - если exp1 = 1