КАЛУЖСКИЙ ФИЛИАЛ
ФЕДЕРАЛЬНОГО ГОСУДАРСТВЕННОГО БЮДЖЕТНОГО
ОБРАЗОВАТЕЛЬНОГО УЧРЕЖДЕНИЯ ВЫСШЕГО ОБРАЗОВАНИЯ
«МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ ИМЕНИ Н.Э. БАУМАНА
(национальный исследовательский университет)»

Факультет «Информатика и управление»

Кафедра «Программное обеспечение ЭВМ, информационные технологии»

Высокоуровневое программирование

Лекция №2. «Ингредиенты Python: числа, строки и переменные. Управляющие конструкции языка Python»

Калуга - 2020

Типизация языка Python

- Python относится к языкам с **неявной сильной динамической типизацией**
- **Неявная** типизация означает, что при объявлении переменной вам *не нужно указывать её тип*, при явной – это делать необходимо (C++)

int a = 1 (C++) a = 1 (Python)

- В случае **динамической типизации** тип переменной определяется непосредственно *при выполнении программы*, в случае **статической** – *на этапе компиляции*
- **Сильная типизация** выделяется тем, что язык не позволяет смешивать в выражениях различные типы и *не выполняет автоматические неявные преобразования*, например нельзя вычесть из строки множество. Языки со **слабой типизацией** выполняют множество неявных преобразований автоматически, даже если может произойти потеря точности или преобразование неоднозначно

Типы данных

В Python есть несколько стандартных типов данных:

- Numbers (числа)
- Strings (строки)
- Lists (списки)
- Dictionaries (словари)
- Tuples (кортежи)
- Sets (множества)
- Boolean (логический тип данных)
- None (неопределенное значение переменной)

Эти типы данных можно, в свою очередь, классифицировать по нескольким признакам:

- изменяемые (списки, словари и множества)
- неизменяемые (числа, строки и кортежи)
- упорядоченные (списки, кортежи, строки и словари)
- неупорядоченные (множества)

Модель данных

- Целочисленное значение 5 – это *объект*.
- *Объект*, в данном случае – это абстракция для представления данных.
- *Данные* – это числа, списки, строки и т.п.
- Каждый объект имеет три атрибута – это **идентификатор, значение и тип**.
- Идентификатор – это уникальный признак объекта, позволяющий отличать объекты друг от друга, а значение – непосредственно информация, хранящаяся в памяти, которой управляет интерпретатор.
- При инициализации переменной, на уровне интерпретатора, происходит следующее:
  - создается целочисленный объект 5 (можно представить, что в этот момент создается ячейка и 5 кладется в эту ячейку);
  - данный объект имеет некоторый идентификатор, значение: 5, и тип: целое число;
  - посредством оператора “=” создается ссылка между переменной b и целочисленным объектом 5 (переменная b ссылается на объект 5).

```
1  b = 5
2  print(id(5))
3  print(id(b))
```

```
1466148848
1466148848
```

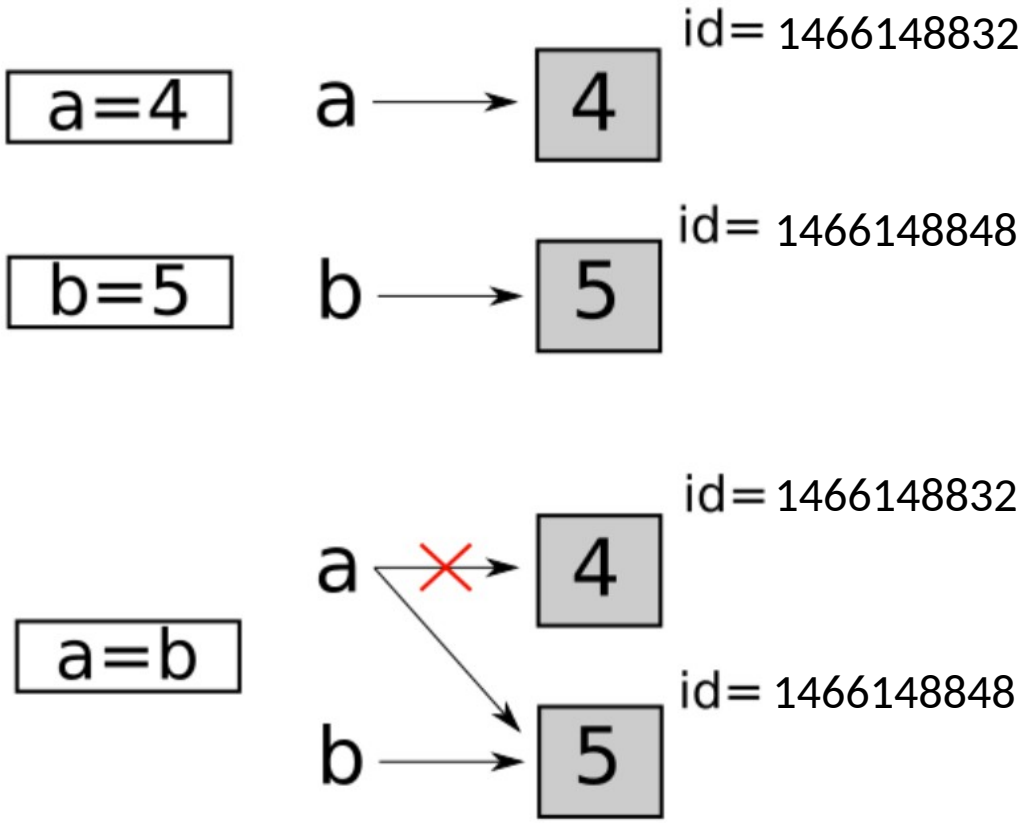
```
1  b = 5
2  print(type(5))
3  print(type(b))
```

```
<class 'int'>
<class 'int'>
```

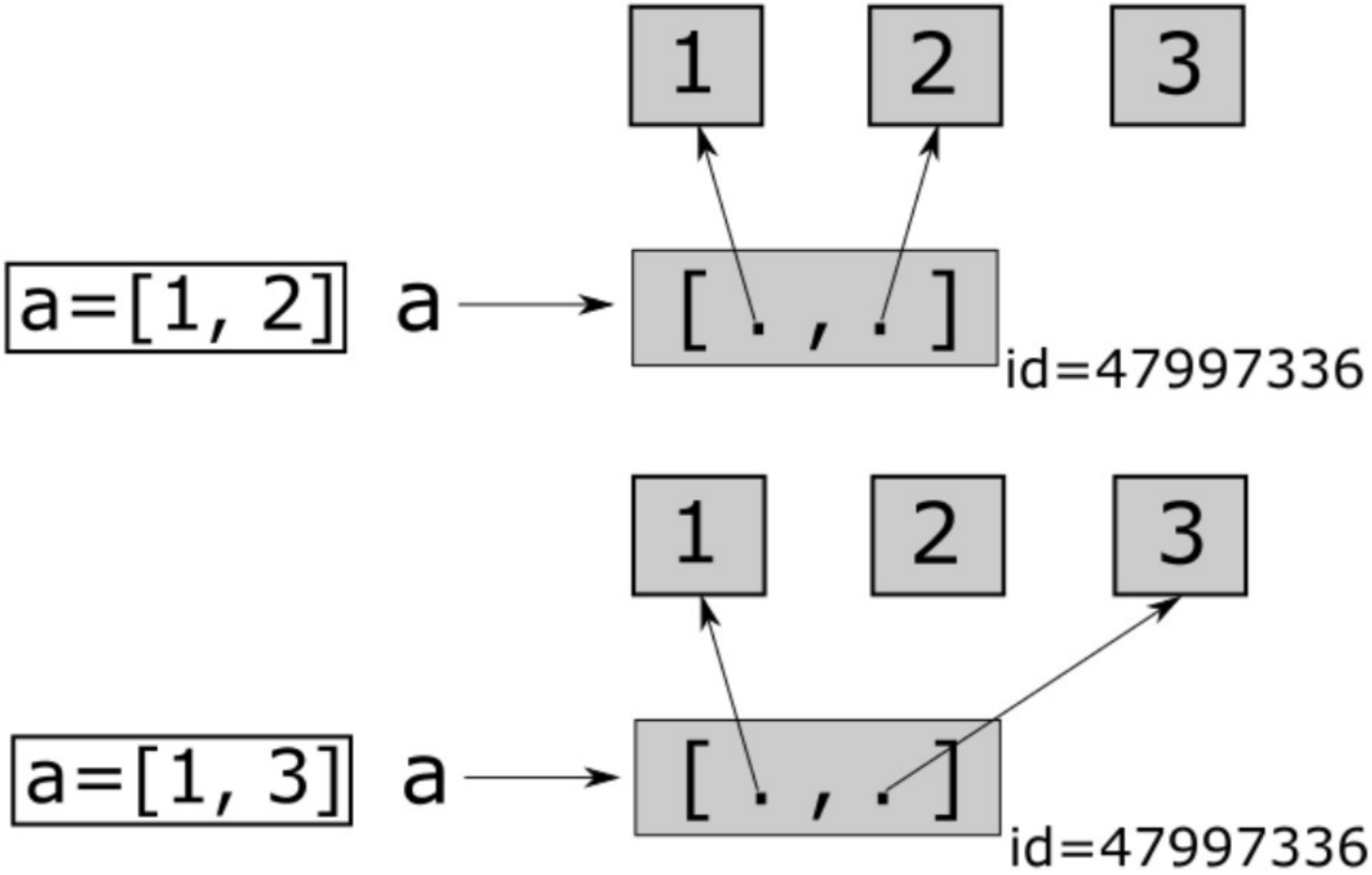
Модель данных

```
1  a = 4
2  b = 5
3  print(id(a))
4  print(id(b))
5  a = b
6  print(id(a))
```

```
1466148832
1466148848
1466148848
```



Изменяемые типы данных



Числа

+, -, *, /	Арифметические операции
**	Возведение в степень
//	Целая часть от деления
%	Остаток от деления
Int(‘11’)	Приведение типа
<, >, <=, >=, ==	Операции сравнения
+=, -=, /=, //=, %=, **=	Сокращенное присвоение
модуль math	Сложные математические функции модуля math

Строки (Strings)

- последовательность символов, заключенная в кавычки;
- **неизменяемый упорядоченный** тип данных.

```
1 s1 = "Hello"
2 s2 = ', world!'
3 s1 + s2
```

```
'Hello, world!'
```

```
1 s1[1]
```

```
'e'
```

```
1 s1[-1]
```

```
'o'
```

Строки

```
1 s3 = s1 + s2
2 s3[0:]
```

```
'Hello, world!'
```

```
1 s3[2:5]
```

```
'llo'
```

```
1 s3[-3:]
```

```
'ld!'
```

```
1 s3[1::2]
```

```
'el,wrd'
```

```
1 s3[::2]
```

```
'Hlo ol!'
```

```
1 len(s3)
```

```
13
```

```
1 s3.upper()
```

```
'HELLO, WORLD!'
```

```
1 s3.lower()
```

```
'hello, world!'
```

```
1 s3.swapcase()
```

```
'hELLO, WORLD!'
```

```
1 s3 = 'hello'
2 s3.capitalize()
```

```
'Hello'
```

Функции для строк

```
1 s3 = "Hello, hello, hEllo, hello"
2 s3.count("hello")
```

```
2
```

```
1 s3.find("lo")
```

```
3
```

```
1 s3.startswith("H")
```

```
True
```

```
1 s3.startswith("h")
```

```
False
```

```
1 s3.endswith("!")
```

```
False
```

```
1 s3.endswith("o")
```

```
True
```

```
1 s3.replace("hello", "student")
```

```
'Hello, student, hEllo, student'
```

Функции для строк

```
1  s4 = "\n(New string)\n"
2  s4
```

```
'\n(New string)\n'
```

```
1  s5 = s4.strip()
2  s5
```

```
'(New string)'
```

```
1  s5.strip("()")  # lstrip(), rstrip()
```

```
'New string'
```

```
1  s3.split()
```

```
['Hello,', 'hello,', 'hEllo,', 'hello']
```

```
1  s3.split(",")
```

```
['Hello', ' hello', ' hEllo', ' hello']
```

```
1  s6 = "h       e         l  l      o"
2  s6.split()
```

```
['h', 'e', 'l', 'l', 'o']
```

Форматирование строк методом format

```
# Специальный символ {} указывает, что сюда подставится значение,
# которое передается методу format. При этом каждая пара фигурных
# скобок обозначает одно место для подстановки.
print('{}'.format(5))
print('{}'.format("string"))
print('{}'.format(['l', 'i', 's', 't']))
```

```
5
string
['l', 'i', 's', 't']
```

```
# вывод данных столбцами одинаковой ширины по 15 символов
# с выравниванием по правой стороне
name, age, address = ['Ivan', 20, 'Kaluga']
print("{:>20} {:>20} {:>20}".format(name, age, address))
```

```
                Ivan                   20                Kaluga
```

```
# выравнивание по левой стороне
print("{:20} {:20} {:20}".format(name, age, address))
```

```
Ivan                                   20 Kaluga
```

```
print("{:.3f} {:b} {:20b}".format(10.0/3, 100, 200))
```

```
3.333 1100100              11001000
```

Условная конструкция if

```
if логическое_выражение:
    инструкции
[elif логическое
выражение:
    инструкции]
[else:
    инструкции]
```

```
age = 22
if age > 21:
    print("Доступ разрешен")
print("Завершение работы")
```

```
age = 22
if age > 21:
    print("Доступ разрешен")
    print("Завершение работы")
```

```
age = 18
if age > 21:
    print("Доступ разрешен")
else:
    print("Доступ запрещен")
```

Условная конструкция if

```
age = 18
if age >= 21:
    print("Доступ разрешен")
elif age >= 18:
    print("Доступ частично разрешен")
else:
    print("Доступ запрещен")
```

```
age = 18
if age >= 18:
    print("Больше 17")
    if age > 21:
        print("Больше 21")
    else:
        print("От 18 до 21")
```

```
age = 18
if age >= 18:
    print("Больше 17")
if age > 21:
    print("Больше 21")
else:
    print("От 18 до 21")
```