Суперкраткая история Python

Основателю Python - Гвидо ван Россум

Разработка была начата в 1989

Python2 вышел в 2000 г. (не поддерживается с 2020)

Python3 вышел в конце 2008 г., он не обладает обратной совместимостью относительно Python2!

Текущая версия 3.10

Типы данных в Python. Изменяемые и

неизменяемые типы. Хранение переменных в памяти

Python – это динамически типизированный язык, тип переменной определяется непосредственно при выполнении программы.

Например, строка:

```
a = 5
```

объявляет переменную b и присваивает ей значение 5.

Python - язык с сильной типизацией, то есть вы не можете складывать например строки и числа, нужно все приводить к одному типу.

In [1]:
```
a = 5
print(type(a))

a = 5.5
print(type(a))

a = 'abc'
print(type(a))

a = [1,2]
print(type(a))
```

```
<class 'int'>
<class 'float'>
<class 'str'>
<class 'list'>
```

In [2]:
```
# попробуем сложить разные типы (int + str)
5 + 'hello'
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-2-1847f954d816> in <module>
      1 # попробуем сложить разные типы (int + str)
----> 2 5 + 'hello'

TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

Типы данных

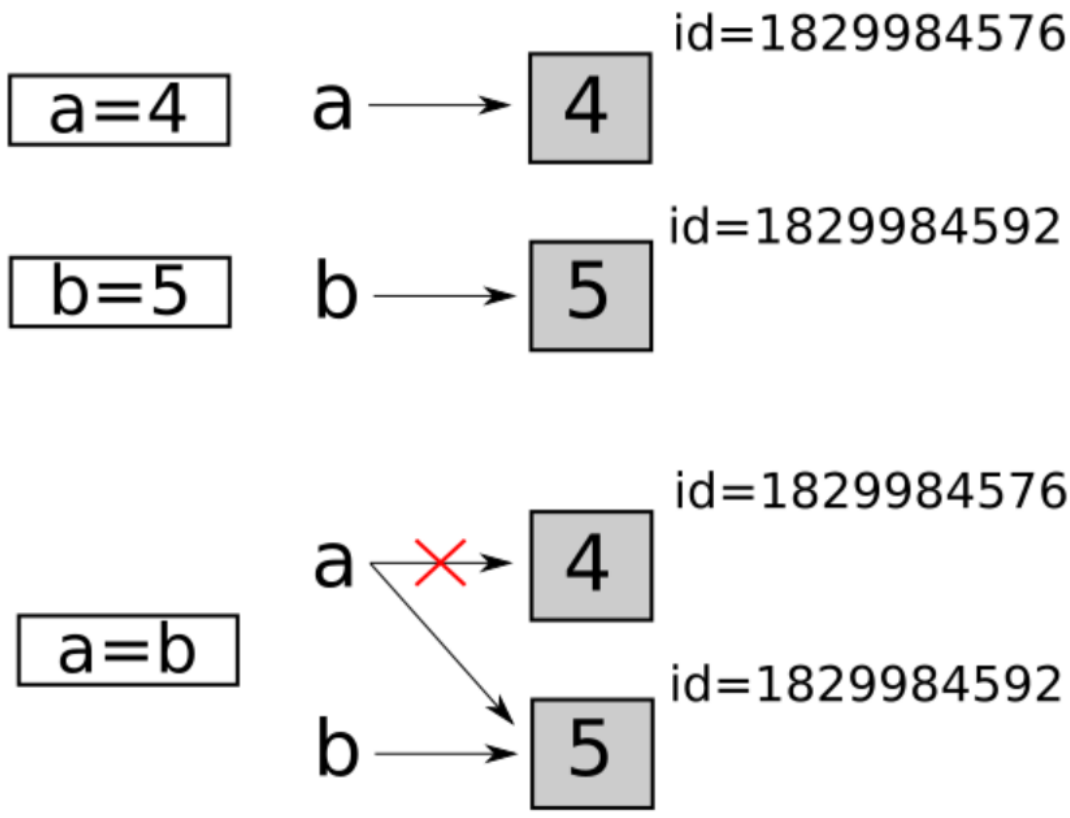
К основным встроенным типам данных в Python относятся:

- **None** (неопределенное значение переменной)
- **Логические переменные** (Boolean Type)
- **NotImplemented** (используется для указания Python, что специальный метод не поддерживает конкретные аргументы, а Python будет пытаться использовать альтернативы, если они доступны)
- **Числа** (Numeric Type)
  - *int* – целое число
  - *float* – число с плавающей точкой

    - *complex* – комплексное число
- **Списки** (Sequence Type)
    - *list* – список
    - *tuple* – кортеж
    - *range* – диапазон
- **Строки** (Text Sequence Type)
    - *str*
- **Бинарные списки** (Binary Sequence Types)
    - *bytes* – байты
    - *bytearray* – массивы байт
    - *memoryview* – специальные объекты для доступа к внутренним данным объекта через protocol buffer
- **Множества** (Set Types)
    - *set* – множество
    - *frozenset* – неизменяемое множество
- **Словари** (Mapping Types)
    - *dict* – словарь

Хранение переменных в памяти

При создании переменной вначале создается **объект**, который имеет **уникальный идентификатор**, **тип** и **значение**, после этого переменная может ссылаться на созданный объект.



Имя переменной не должно совпадать с ключевыми словами интерпретатора Python. Список ключевых слов можно получить непосредственно в программе, для этого нужно подключить модуль keyword и воспользоваться командой keyword.kwlist.

In [3]:
```
import keyword
print("Python keywords: ", keyword.kwlist)
```

```
Python keywords:  ['False', 'None', 'True', 'and', 'as', 'assert', 'async', 'await',
'break', 'class', 'continue', 'def', 'del', 'elif', 'else', 'except', 'finally', 'fo
r', 'from', 'global', 'if', 'import', 'in', 'is', 'lambda', 'nonlocal', 'not', 'or',
'pass', 'raise', 'return', 'try', 'while', 'with', 'yield']
```

Проверить является или нет идентификатор ключевым словом можно так:

In [4]:
```
print(keyword.iskeyword("try"))

print(keyword.iskeyword("b"))
```

```
True
False
```

In [5]:
```
a = 4789
id(a)
```

Out[5]:
```
1270925550480
```

In [6]:
```
b = 4789
id(b)
```

Out[6]:
```
1270925551088
```

In [7]:
```
a = 19.5

print('идентификатор: ', id(a))
print('тип:           ', type(a))
print('значение:      ', a)
```

```
идентификатор:  1270925550288
тип:            <class 'float'>
значение:       19.5
```

Операция `*3` при создании объекта копирует ссылки. Это видно на следующем примере:

In [8]:
```
a = [[]] * 3
print(a)

a[0].append(3)
print(a)
```

```
[[], [], []]
[[3], [3], [3]]
```

Изменяемые и неизменяемые типы данных

В Python существуют изменяемые и неизменяемые типы.

К **неизменяемым** (*immutable*) типам относятся:

- целые числа (*int*)
- числа с плавающей точкой (*float*)

- комплексные числа (*complex*)
- логические переменные (*bool*)
- кортежи (*tuple*)
- строки (*str*)
- неизменяемые множества (*frozen set*).

К **изменяемым** (mutable) типам относятся:

- списки (*list*)
- множества (*set*)
- словари (*dict*)

Неизменяемые типы

Неизменяемость типа данных означает, что созданный объект больше не изменяется. При изменении значения происходит создание нового объекта, на который теперь будет ссылаться переменная.

In [9]:
```
a = 2
print(id(a))

a = 4
print(id(a))
```

```
140707253659472
140707253659536
```

Если выполнить операцию присваивания

```
a = b
```

то переменная `a` будет ссылаться на тот же объект, что и переменная `b`.

Узнать, ссылаются ли переменные на один и тот же объект, можно при помощи операторов тождественности:

- `is`
- `is not`

In [10]:
```
a = 16
print('id(a) = ', id(a))

b = 2.2
print('id(b) = ', id(b))

a = b

print('id(a) = ', id(a))
print(a is b)
```

```
id(a) =  140707253659920
id(b) =  1270925550672
id(a) =  1270925550672
True
```

Так как строки - также неизменяемые объекты, при попытке изменить какой-нибудь символ в строке произойдет ошибка:

In [11]:
```
s = "абракадабра"
s[3] = 'ы'
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-11-8c71338b2bc7> in <module>
      1 s = "абракадабра"
----> 2 s[3] = 'ы'

TypeError: 'str' object does not support item assignment
```

Изменяемые типы

Изменяемыми объектами являются списки, множества и словари, и их можно менять:

- Менять элементы
- Добавлять/удалять элементы

В примере ниже создан список, а потом изменен второй элемент.

В качестве данных списка выступают не объекты, а отношения между объектами. Т.е. в переменной a хранятся ссылки на объекты содержащие числа 1 и 3, а не непосредственно сами эти числа.

In [12]:
```
# Создаем список
a = [1, 2]

print(a)
print('a:    ', id(a))
print('a[1]: ', id(a[1]))
print()

# Изменяем один элемент
a[1] = 3

print(a)
print('a:    ', id(a))
print('a[1]: ', id(a[1]))
print()

# Добавляем еще один элемент в список
a.append(4)

print(a)
print('a:    ', id(a))
```

```
[1, 2]
a:     1270925494464
a[1]:  140707253659472

[1, 3]
a:     1270925494464
a[1]:  140707253659504

[1, 3, 4]
a:     1270925494464
```