Практическая работа №1. Типы данных в Python

В Python типы данных можно разделить на **встроенные в интерпретатор *(built-in*)** и не встроенные, которые можно использовать при импортировании соответствующих модулей.
К основным встроенным типам относятся:

1. *None* (неопределенное значение переменной)
2. Логические переменные (*Boolean Type*)
3. Числа (*Numeric Type*)
    1. *int* – целое число
    2. *float* – число с плавающей точкой
    3. *complex* – комплексное число
4. Списки (*Sequence Type*)
    1. *list* – список
    2. *tuple* – кортеж
    3. *range* – диапазон
5. Строки (*Text Sequence Type*)
    1. *str*
6. Бинарные списки (*Binary Sequence Types*)
    1. *bytes* – байты
    2. *bytearray* – массивы байт
    3. *memoryview* – специальные объекты для доступа к внутренним данным объекта через protocol buffer
7. Множества (*Set Types*)
    1. *set* – множество
    2. *frozenset* – неизменяемое множество
8. Словари (*Mapping Types*)
    1. *dict* – словарь

Модель данных
Рассмотрим как создаются объекты в памяти, их устройство, процесс объявления новых переменных и работу операции присваивания.

Чтобы объявить переменную и сразу ее инициализировать: необходимо написать её имя, потом поставить знак равенства и значение, с которым эта переменная будет создана. Например:

```
b = 5
```

объявляет переменную b и присваивает ей значение 5.

Целочисленное значение 5 в рамках языка Python по сути своей является объектом
Объект – это абстракция для представления данных,
данные – это числа, списки, строки и т.п.

При этом, под данными следует понимать как непосредственно сами объекты, так и отношения между ними

Каждый объект имеет три атрибута – это идентификатор, значение и тип. Идентификатор – это уникальный признак объекта, позволяющий отличать объекты друг от друга,
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 идентификатор **b**, значение: **5**, и тип: **целое число**;

- посредством оператора “=” создается ссылка между переменной b и целочисленным объектом 5 (переменная b ссылается на объект 5).

!!! Имя переменной не должно совпадать с ключевыми словами интерпретатора Python.
Список ключевых слов Записать в тетрадь! ***Python keywords: ['False', 'None', 'True', 'and', 'as', 'assert', 'break', 'class', 'continue', 'def', 'del', 'elif', 'else', 'except', 'finally', 'for', 'from', 'global', 'if', 'import', 'in', 'is', 'lambda', 'nonlocal', 'not', 'or', 'pass', 'raise', 'return', 'try', 'while', 'with', 'yield']***
! Можно непосредственно в программе проверить, является ли имя переменной ключевым словом? Для этого нужно подключить модуль *keyword* и воспользоваться командой *keyword.kwlist.*

```
>>> import keyword
>>> print("Python keywords: ", keyword.kwlist)
```

Проверить является или нет идентификатор ключевым словом можно так:

```
>>> keyword.iskeyword("try")
True
>>> keyword.iskeyword("b")
False
```

Для того, чтобы посмотреть на объект с каким идентификатором ссылается данная переменная, можно использовать функцию ***id().***

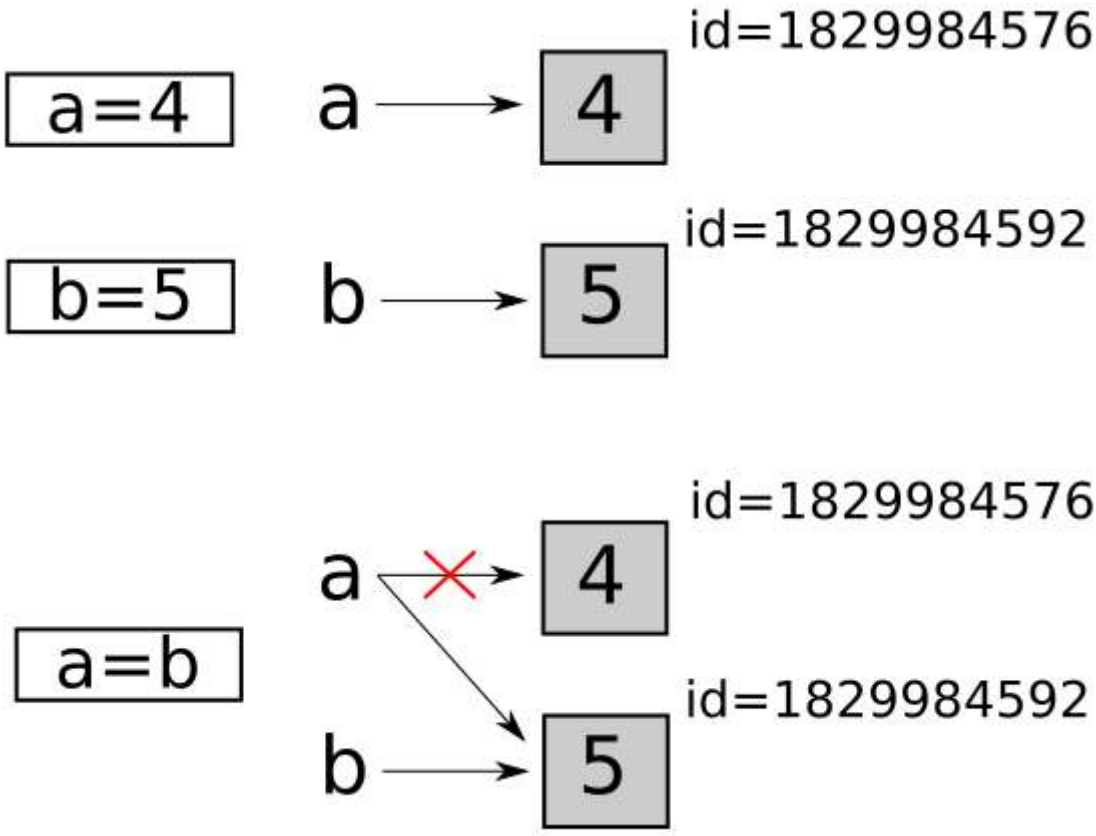
```
>>> a = 4
>>> b = 5
>>> id(a)
1829984576
>>> id(b)
1829984592
>>> a = b
>>> id(a)
1829984592
```

Как видно из примера, **идентификатор** – это некоторое целочисленное значение, посредством которого уникально адресуется объект. Изначально переменная *a* ссылается на объект 4 с идентификатором 1829984576, переменная *b* – на объект с *id* = 1829984592. После выполнения операции присваивания *a* = *b*, переменная *a* стала ссылаться на тот же объект, что и *b*.

Задание:

1) **Отработать Пайтоне все методы по теме**
2) **Переписать в конспект на память**
3) **Оформить в виде отчета в Ворде(скрины)**
4) **Сохранить отчет в PDFформате с титульным листом**
5) **Отправить до 9-00 следующего дня**

С уважением Баян Е



Тип переменной можно определить с помощью функции *type()*.
Пример использования приведен ниже.

```
>>> a = 10
>>> b = "hello"
>>> c = (1, 2)
>>> type(a)
<class 'int'>
>>> type(b)
<class 'str'>
>>> type(c)
<class 'tuple'>
```

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*) и неизменяемые множества (*frozen set*).
К **изменяемым** (*mutable*) типам относятся: списки (*list*), множества (*set*), словари (*dict*).
Как уже было сказано ранее, при создании переменной, вначале создается объект, который имеет уникальный идентификатор, тип и значение, после этого переменная может ссылаться на созданный объект.
Неизменяемость типа данных означает, что созданный объект больше не изменяется. Например, если мы объявим переменную *k* = 15, то будет создан объект со значением 15, типа *int* и идентификатором, который можно узнать с помощью функции *id()*.

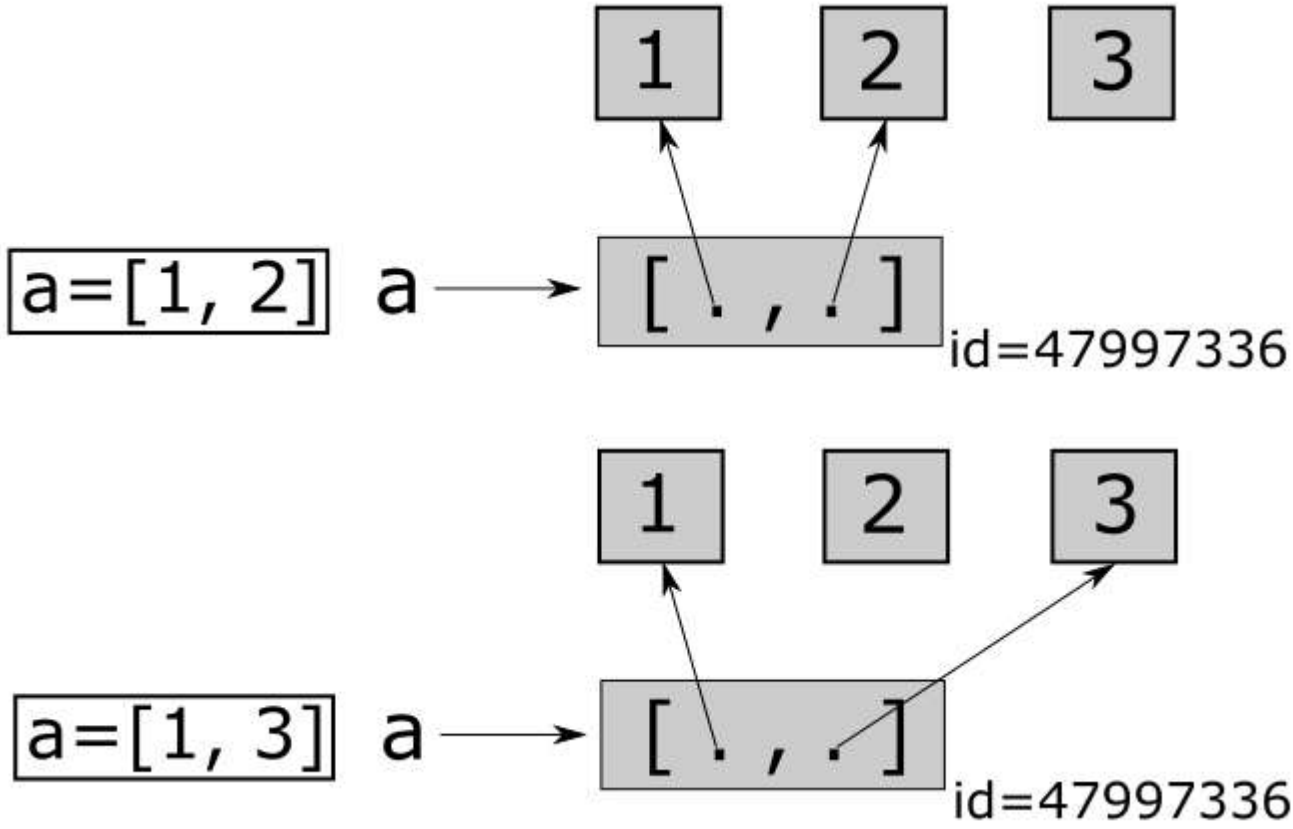
```
>>> k = 15
>>> id(k)
1672501744
>>> type(k)
```

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

Объект с *id* = 1672501744 будет иметь значение 15 и изменить его уже нельзя.
Если тип данных изменяемый, то можно менять значение объекта.
Например, создадим список [1, 2], а потом заменим второй элемент на 3.

```
>>> a = [1, 2]
>>> id(a)
47997336
>>> a[1] = 3
>>> a
[1, 3]
>>> id(a)
47997336
```

Как видно, объект на который ссылается переменная *a*, был изменен. Это можно проиллюстрировать следующим рисунком.



В рассмотренном случае, 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.