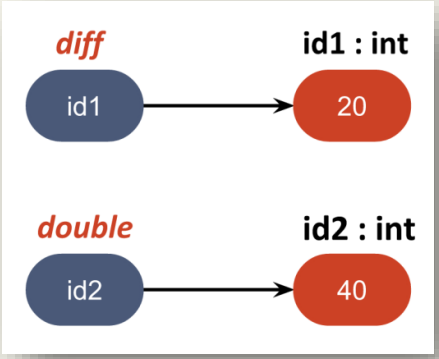
В момент выполнения присваивания `cel = 26` в памяти компьютера создается объект, расположенный по некоторому адресу[9] (условно обозначим его как `id1`), имеющий значение 26 целочисленного типа `int`. Затем создается переменная с именем `cel`, которой присваивается адрес объекта `id1`. Переменные в Python содержат адреса объектов или можно сказать, что переменные ссылаются на объекты. Постоянно сохраняя в голове эту модель, для упрощения будем говорить, что переменная содержит значение.

Вычисление следующего выражения в итоге приведет к присваиванию переменной `cel` значения 72, т.е. сначала вычисляется правая часть, затем результат присваивается левой части.

```
>>> cel = 26 + 46
>>> cel
72
>>>
```

Рассмотрим чуть более сложный пример. Вместо переменной `diff` подставится целочисленное значение 20:

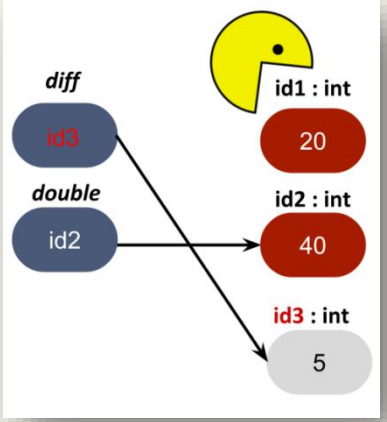
```
>>> diff = 20
>>> double = 2 * diff
>>> double
40
>>>
```



По окончании вычислений память для Python будет иметь следующий вид:

Продолжим вычисления. Присвоим переменной `diff` значение 5 и посмотрим содержимое переменных `double` и `diff`.

```
>>> diff = 5
>>> double
40
>>> diff
5
>>>
```



В момент присваивания переменной `diff` значения 5 в памяти создастся объект по адресу `id3`, содержащий целочисленное значение 5. После этого изменится содержимое переменной `diff`, вместо адреса `id1` туда запишется адрес `id3`. Также Python увидит, что на объект по адресу `id1` больше никто не ссылается и поэтому удалит его из памяти (произведет автоматическую сборку мусора).

Внимательный читатель заметил, что Python не изменяет существующие числовые объекты, а создает

---

[9] ***Информация для опытных программистов.*** Функция `id` возвращает идентификатор объекта, переданного в качестве аргумента функции. В реализации CPython возвращаемое число является адресом объекта в памяти.