In [2]:
```
num = 13
```

In [4]:
```
num.__add__(2)
```

Out[4]: 15

In [13]:
```
print(dir(num))
```

```
['__abs__', '__add__', '__and__', '__bool__', '__ceil__', '__class__', '__delattr__', '__dir
__', '__divmod__', '__doc__', '__eq__', '__float__', '__floor__', '__floordiv__', '__format_
_', '__ge__', '__getattribute__', '__getnewargs__', '__gt__', '__hash__', '__index__', '__in
it__', '__init_subclass__', '__int__', '__invert__', '__le__', '__lshift__', '__lt__', '__mo
d__', '__mul__', '__ne__', '__neg__', '__new__', '__or__', '__pos__', '__pow__', '__radd__',
'__rand__', '__rdivmod__', '__reduce__', '__reduce_ex__', '__repr__', '__rfloordiv__', '__rl
shift__', '__rmod__', '__rmul__', '__ror__', '__round__', '__rpow__', '__rrshift__', '__rshi
ft__', '__rsub__', '__rtruediv__', '__rxor__', '__setattr__', '__sizeof__', '__str__', '__su
b__', '__subclasshook__', '__truediv__', '__trunc__', '__xor__', 'bit_length', 'conjugate',
'denominator', 'from_bytes', 'imag', 'numerator', 'real', 'to_bytes']
```

In [2]:
```
print(dir("строка"))
```

```
['__add__', '__class__', '__contains__', '__delattr__', '__dir__', '__doc__', '__eq__', '__f
ormat__', '__ge__', '__getattribute__', '__getitem__', '__getnewargs__', '__gt__', '__hash_
_', '__init__', '__init_subclass__', '__iter__', '__le__', '__len__', '__lt__', '__mod__',
'__mul__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__rmod__', '__rm
ul__', '__setattr__', '__sizeof__', '__str__', '__subclasshook__', 'capitalize', 'casefold',
'center', 'count', 'encode', 'endswith', 'expandtabs', 'find', 'format', 'format_map', 'inde
x', 'isalnum', 'isalpha', 'isdecimal', 'isdigit', 'isidentifier', 'islower', 'isnumeric', 'i
sprintable', 'isspace', 'istitle', 'isupper', 'join', 'ljust', 'lower', 'lstrip', 'maketran
s', 'partition', 'replace', 'rfind', 'rindex', 'rjust', 'rpartition', 'rsplit', 'rstrip', 's
plit', 'splitlines', 'startswith', 'strip', 'swapcase', 'title', 'translate', 'upper', 'zfil
l']
```

```
typedef struct _object {
    _PyObject_HEAD_EXTRA
    Py_ssize_t ob_refcnt;
    struct _typeobject *ob_type;
} PyObject;
```

```
typedef struct _object {
    _PyObject_HEAD_EXTRA
    Py_ssize_t ob_refcnt;          // Счетчик ссылок
    struct _typeobject *ob_type;  // Указатель на тип объекта
} PyObject;


typedef struct {
    PyObject ob_base;
    Py_ssize_t ob_size;  // Кол-во элементов в переменной части
} PyVarObject;


typedef struct {
    PyObject ob_base;
    Py_ssize_t ob_size; // Кол-во элементов в переменной части
} PyVarObject;


#define PyObject_HEAD  PyObject ob_base;
```

```
#define PyObject_VAR_HEAD  PyVarObject ob_base;


typedef struct PyMyObject {
   PyObject_HEAD
   ...
}
```

или

```
typedef struct PyMyObject {
   PyObject_VAR_HEAD
   ...
}
```

Практически все в Python наследуется от PyObject, является объектом и может быть присвоено переменной и быть передано в качестве аргумента в функцию! Не только базовые типы, такие как int, float, bool, str и т.д., но также и функции и даже модули, содержащие наш код на Python.

Теперь вы знаете, что почти все в Python имплементировано как объект, у каждого такого объекта есть счетчик ссылок и описание типа, и в конечном итоге у объекта определенного типа есть масса методов и атрибутов, которые мы с вами видели, например, вызывая функцию dir с целочисленным объектом в качестве аргумента. Также раскрыта тайна почему я постоянно произносил что мы связываем имя переменной с объектом, а не просто что мы присваиваем переменной значение. Я хотел подчеркнуть особенность объектной структуры Python, о которой мы только что поговорили.