Типы данных

В Python все типы данных можно поделить на изменяемые (mutable) и неизменяемые (immutable). Изменяемые типы данных передаются по ссылке, а неизменяемые по значению.

Неизменяемые типы

Нельзя изменить значение переменной. Ее можно только переопределить, т. е. сделать так, чтобы переменная ссылалась на другой объект памяти. Все неизменяемые типы данных передаются по значению.

тип	описание
`bool`	логический тип `a, b = True, False`
`int`	32-битное целое число (-2 147 483 647 до 2 147 483 647)
`float`	32-битное число с плавающей запятой (6-7 значимых цифр)
`double`	32-битное число с плавающей запятой (15–6 значимых цифр)
`str`	последовательность символов `my_str='Max'`
`tuple`	кортеж (неизменяемый список) `my_tuple=(1,'Max','123')`
`frozenset`	Неизменяемое множество `my_frozenset=frozenset({1,'Max'})`

Изменяемые типы

Можно изменить значение переменной. Все изменяемые типы данных передаются по ссылке.

тип	описание
`list`	список — последовательность элементов `my_list=[1,'Max','123']`
`set`	множество — коллекция уникальных элементов `my_set={1,'Max','123'}`
`dict`	словарь — коллекция элементов вида «ключ: значение» `my_dict={1:'Max','email':'my_email','age':25}`
`function`	функция `def my_func(): pass`
`my_class`	экземпляр (объект) пользовательского класса `my_obj=my_class()`

Получить уникальный идентификатор объекта, на который ссылается переменная

```
id(<имя_переменной>)
```

Получить тип объекта, на который ссылается переменная

```
type(<имя_переменной>)
```

Python – это язык со строгой, динамической типизацией

«Динамическая» означает, что в одной переменной сначала могут хранится данные одного типа, а потом другого

```
a=3 # в переменной лежат данные типа int

a='Привет мир!' # теперь в переменной лежат
                # данные типа str
```

«Строгая» означает, что язык не делает неявных преобразований типов

```
a=3
b='Привет мир!'
c=a+b # будет ошибка. Python сам не
      # приведет данные к одному типу
```

Сделать обычную копию переменной с изменяемым типом данных

```
import copy
copy_obj=copy.copy(my_obj)
```

Сделать глубокую копию переменной с изменяемым типом данных

```
import copy
copy_obj=copy.deepcopy(my_obj)
```