STEP ACADEMY

ЗАНЯТИЕ №3

Переменные и типы данных

Базовый уровень абстракции

В рабочих программах редко встречаются конкретные данные. Это почти всегда данные, полученные от пользователя или других источников, которые мы должны как-то обработать, чтобы выдать ответ.

Более того, конкретные данные, прописанные *явно* в коде программы (константы), часто являются источниками проблем и «глюков».

Поэтому в программах работают с абстрактными данными, используя **переменные**.

Переменные

В языке Python переменные являются своего рода «стикерами» - **именованными ссылками** на область памяти, где хранится объект.

Для создания переменных используется оператор присваивания: **=**

```
current_price = 3500
```

Эта запись обрабатывается интерпретатором так:

1) В памяти создаётся объект типа ***int*** (целое число) со значением 3500;

2) В текущей области видимости программы создаётся ссылка с именем current_price, содержащая *адрес этого объекта в памяти*;

3) Счетчик ссылок этого объекта становится равным 1.

Переменные

```
best_price = current_price
```

Так как в правой части выражения находится переменная, то создания нового объекта не происходит. Вместо этого:

1) В текущей области видимости программы создаётся ещё одна ссылка с именем best_price, содержащая *адрес того же объекта,* на который указывает ссылка current_price;

2) Счетчик ссылок этого объекта увеличивается на 1.

Сборка мусора

Когда выполнение программы выходит из области видимости переменных, или им присваиваются новые значения, происходит уменьшение значения счётчика ссылок для того объекта, на который указывали эти переменные.

Как только счетчик ссылок объекта становится равным 0, это значит, что он уже нигде в программе не используется и является «мусором». Такие объекты автоматически удаляются из памяти.

Варианты присваивания

1) Присваивание результата выполнения функции:

```
result = input("Введите число: ")
```

2) Позиционное присваивание:

```
hours, minutes, seconds = 20, 55, 17
```

3) Каскадное присваивание:

```
title = description = author = ""
```

Типы данных

При создании объекта тип его данных Python определяет автоматически:

Тип данных	Класс	Описание
Числа	int, float, complex	Содержит числовые значения.
Строки	str	Содержит последовательность символов.
Списки	list, tuple, range	Содержит последовательность элементов.
Словари	dict	Содержит данные в виде пары ключ-значение.
Логический	bool	Содержит либо True, либо False.
Множество	set, frozeenset	Содержит последовательность уникальных элементов.

Преобразование типов

Чтобы узнать тип объекта, можно воспользоваться функцией type():

```
>>> type(3.14)
<class 'float'>
```

Для преобразования типов используются функции, имена которых совпадают с именами типов:

```
>>> string_pi = str(3.14)
>>> type(string_pi) is str
True
```

Ошибки преобразования типов

Не все типы можно преобразовать в другие:

```
>>> list(3.14)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'float' object is not iterable
```

Не все значения могут быть преобразованы:

```
>>> int("seven")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: invalid literal for int() with base 10: 'seven'
```

Операции и операторы

Оператор — это набор символов (идентификатор) обозначающий какую-то операцию. Например, оператор `+`

Операция — это действие, которое выполняется с аргументами оператора. В зависимости от типов аргументов, один и тот же оператор будет выполнять разную операцию:

```
4 + 5                 # 9
"desk" + "top"        # "desktop"
```

Python не делает автоматическое преобразование типов аргументов!

Операции над числовыми типами

Оператор	Операция
+	сложение
-	вычитание
*	умножение
/	деление
//	целочисленное деление
%	остаток от деления
**	возведение в степень

В результате операции создаётся новый объект, которому может быть присвоена переменная.

Составные операторы

Выражение	Эквивалент
`a += b`	`a = a + b`
`a -= b`	`a = a - b`
`a *= b`	`a = a * b`
`a /= b`	`a = a / b`
`a //= b`	`a = a // b`
`a %= b`	`a = a % b`
`a **= b`	`a = a ** b`

Конец