InnoDom

Переменные и типы данных

Переменная

это именованная область памяти, в которой хранится информация. Каждая переменная содержит тип и название.

Тип переменной определяется в зависимости от того, какие данные в ней находятся.

Базовые типы в Python

Данные	Название (англ)	Название в Python
Целочисленный тип	Integer	`int`
Вещественный тип	Float	`float`
Строковый тип (текст)	String	`str`
Нулевое поведение	None Type	`NoneType`
Логический тип	Boolean	`bool`

Формула создания переменной:

название переменной = данные

```
int_variable = 45
float_variable = 45.45
str_variable = "Hello world!"
none_variable = None
bool_variable = True # False
```

! При создании переменной тип не указывается явно, а определяется сам под "капотом", в зависимости от того, какие данные записываются в переменную. Такой процесс называется **динамической типизацией**.

Правильные названия переменных

- Переменные принято записывать в формате `snake_case`, т.е. если название переменной состоит из нескольких слов, то принято писать их через знак **нижнего подчёркивания**;
- Переменная всегда начинается с **буквы** (`case`) или **нижнего подчеркивания** (`_case`);
- Переменная **не может** начинаться с **цифры**

Правильные названия переменных

- Переменная может содержать только **буквенно-цифровые** символы и подчеркивание (`Az`, `0-9` и `_`);
- Переменные **чувствительны к регистру** (`case`, `Case` и `CASE` — три разные переменные);
- Переменная **не должна** совпадать с зарезервированным **ключевым словом** (это слова, которые заняты названиями конструкций или функций Python)

Функция `print()`

Консольные приложения - это программы, взаимодействие с которыми происходит через консоль.
Функции **принимают** входные параметры (их ещё называют аргументами функции) и **выдают результат**. Если функция **не выдает** результат, то её называют **процедурой**.

! Параметры функции передаются в скобки. В каждой функции параметры что-то значат. Например: `print(param1)`

Функция `print()`

Функция `print()` принимает в себя обязательный параметр (переменную, которую нужно вывести в консоль), выводит его в консоль и **ничего не возвращает**.

Также функция `print()` может принимать **несколько параметров** через запятую, в консоль они выводятся также через запятую.
Например: `print(param1, param2, param3, param4)`

Функция `type()`

Функция `type()` также **принимает** в себя один параметр - переменную и **возвращает** результат – тип переменной.
Функции, которые возвращают результат, можно присваивать переменным.

Функция `type()`

Результат `int` – это и есть тип переменной. Также функции могут принимать не только переменные, а сразу данные.

```
num1 = 100
type1 = type(num1)
print(type1)
```

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

Функция `type()`

Данные нигде не храним, а передаём их сразу в функцию и получаем результат её выполнения. Так передавать параметры можно в любые функции.

```
print(type(100))
```

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

Числа

Арифметические операции

Оператор	Описание
+	Используется для сложения двух чисел
-	Используется для вычитания второго числа из первого
/	Используется для деления первого числа на второе: 5/2 = 2.5
*	Используется для умножения двух чисел
%	Используется для нахождения остатка от деления двух чисел: 5%2 - остаток 1, 15%11 - остаток 4
**	Возведение первого числа в степень второго
//	Используется для целочисленного деления: 5//2 = 2

! Арифметические операции производятся над числами типа `int` и `float`. Если в выражении хотя бы одна часть типа `float`, то результат будет тоже типа `float`.

Передача аргументов в функцию

```
num1 = 10
num2 = 5
print(num1 + num2)
num3 = num1 - num2
print(num3)
print(num1 * num2)
print(10 / 2)
print(num1 % num2)
print(12 % 5)
print(12 // 5)
print(2 ** 2)
```

С помощью переменных: сразу передаём данные и данные с переменными.

! Если какой-то результат в последующем нужно использовать **повторно**, то можно заключить его в **переменную**. Если какое-то действие нужно совершить **один раз**, то хранить информацию в переменной бессмысленно.

Числа

Операторы присваивания

Оператор	Описание
=	Присваивает значение выражения переменной
+=	Увеличивает значение переменной на выражение
-=	Уменьшает значение переменной на выражение
/=	Делит значение переменной на выражение
*=	Умножает значение переменной на выражение
%=	Делит значение переменной на выражение и ищет остаток
**=	Возводит переменную в степень выражения
//=	Нацело делит значение переменной на выражение

Способы присваивания переменных

```
num2 = 10
num1 = num2
print(num1, num2)
```

```
num1 = num2 = 100
print(num1, num2)
```

```
num1, num2 = 10, 100
print(num1, num2)
```

```
num1 = 5
num2 = 10
print(num1)
num1 += 5
print(num1)
num1 -= 2
print(num1)
num1 /= 4
print(num1)
num1 *= num2
print(num1)
num1 %= 9
print(num1)
num1 *= 20
print(num1)
num1 //= 11
print(num1)
```

! Операторы присваивания работают только с переменными.

Если попробовать присвоить выражению 4 выражение 2 (4=2), то будет **ошибка**, т.к. такая запись не имеет смысла.
Поэтому можно присваивать переменной **только данные** или другие **не пустые переменные** одного типа.

Логический тип

Логический тип содержит два утверждения:

- **True** (Истина / Да / 1);
- **False** (Ложь / Нет / 0).

Такой тип необходим в сравнении значений, условных конструкциях, создании флагов.

```
bool1 = True

bool2 = False
```

NoneType

Тип `NoneType` - нулевое или нейтральное поведение. Переменные такого типа существуют, но **ничего не хранят**.
Для создания такой переменной необходимо присвоить ей значение `None`.

```
none1 = None
```

Обычно такие выражения используются для проверки результатов, если он не `None`, то выполняем программу дальше.

NoneType

Строки используются для работы с текстом или символами.

```
str1 = 'Hello world!'
print(str1)
str2 = "Hello world!"
print(str2)
str3 = """Hello
world!"""
str4 = '''


1234
'''
print(str3, str4)
```

Строки могут записываться в **двойных** или **одинарных** кавычках.

Если строка должна сохранять исходное форматирование (переход к следующей строчке, табуляция), то она записывается в **тройных** кавычках.

Типизация данных

У каждой переменной есть свой тип. Если задача состоит в смене одного типа на другой, поможет **типизация данных**.

Если нужно что-то перевести к строке, то можно вызвать функцию `str()`, если в целое число - `int()`, вещественное число - `float()`, если в логический тип - `bool()`.
Таким образом можно проводить операции разных типов на данных, поддающихся типизации данных.

Типизация данных

```
num1 = 100
print(type(num1))
print(type(str(num1)))
print(type(int(num1)))
str1 = "True"
print(type(str1))
print(type(bool(str1)))
num2 = 1
print(type(num2))
print(type(bool(num2)))
print(type(float(num2)))
```

Типизация данных

! Если строку, например, “hello” перевести в число или иной тип, то сработает исключение `ValueError`, т.к. теряется смысловая нагрузка, ведь текст невозможно перевести в число.

Если в строке будут только цифры, например, `"123 "`, то можно перевести их в число и применять к ним числовые операции.

Функция `input()`

Функция `input()` позволяет осуществлять **консольный ввод данных** в вашу программу, для этого требуется вызвать переменную.
Функция принимаем аргумент - текст помощника в консоли и возвращает пользовательский ввод.

```
str1 = input("Введите любую строку: ")
print(str1)
```

! Результат возвращаемого значения всегда `str`.

Комментарии

В коде можно использовать *комментарии*. Они нужны для пометки кода либо комментирования самого кода.

- Чтобы закомментировать код нужно поставить # перед строкой.
- Для того чтобы сделать многострочный комментарий, нужно заключать код в **тройные кавычки** (""").

```
# num1 = 10
# num2 = 5
# print(num1 + num2)
# num3 = num1 - num2
# print(num3)
# print(num1 * num2)
# print(10 / 2)
# print(num1 % num2)
# print(12 % num3)
# print(12 // num3)
# print(2 ** 2)
```

```
"""
num1 = 10
num2 = 5
print(num1 + num2)
num3 = num1 - num2
print(num3)
print(num1 * num2)
print(10 / 2)
print(num1 % num2)
print(12 % num3)
print(12 // num3)
print(2 ** 2)
"""
```