Лекция 5 - Обзор переменных и основных типов данных в Python

Обзор переменных и основных типов данных в Python: как их использовать и объявлять

Что такое переменные в Python?

Переменная в Python — это имя, которое связывается с объектом в памяти компьютера. Переменные служат для хранения данных и обращения к ним в процессе выполнения программы. Главная особенность Python заключается в том, что переменные не требуется объявлять явно — они создаются автоматически при первом присваивании значения.

Правила именования переменных

- При создании переменных в Python необходимо соблюдать следующие правила:
- Имя переменной должно начинаться с буквы (a-z, A-Z) или символа подчеркивания (_)
- После первого символа могут следовать буквы, цифры (0-9) или символы подчеркивания
- Регистр символов имеет значение: переменные name, Name и NAME будут различными
- Нельзя использовать зарезервированные слова Python (например, if, for, while)
- Рекомендуется использовать осмысленные имена переменных

Правильные имена переменных

```
user_name = "Иван"
age = 25
_private_var = "секретные данные"
counter1 = 0
```

Неправильные имена переменных

```
1name = "ошибка"  начинается с цифры
class = "ошибка"  зарезервированное слово
```

Основные типы данных в Python

Числовые типы данных

```
int (integer) — целые числа Используются для представления целых чисел без
дробной части.
positive_number = 42
negative_number = -17
zero = 0
large_number = 1000000
float (floating-point) — числа с плавающей точкой Представляют вещественные
числа с дробной частью.
pi = 3.14159
temperature = -0.5
scientific_notation = 1.23e-4  # 0.000123
complex (комплексные числа) — комплексные числа Состоят из действительной и
мнимой частей, записываются в виде a + bj.
complex_num1 = 3 + 4j
complex_num2 = 2 - 5j
only_imaginary = 7j
```

Строковый тип данных (string)

Строки представляют последовательности символов, заключенные в кавычки. Python поддерживает одинарные, двойные и тройные кавычки.

```
single_quotes = 'Привет, мир!'
double_quotes = "Python программирование"
triple_quotes = """Многострочная
строка в Python"""
empty_string = ""
```

Логический тип данных (bool)

Логический тип принимает только два значения: True (истина) или False (ложь). Используется для выполнения логических операций и условных проверок.

```
is_student = True
is_completed = False
has_permission = True
```

Коллекции данных

Списки (list) — упорядоченные изменяемые коллекции Могут содержать элементы различных типов данных.

```
numbers = [1, 2, 3, 4, 5]
mixed_list = [1, "текст", True, 3.14]
empty_list = []
nested_list = [[1, 2], [3, 4], [5, 6]]
```

Кортежи (tuple) — упорядоченные неизменяемые коллекции После создания элементы кортежа нельзя изменить.

```
coordinates = (10, 20)
colors = ("красный", "зеленый", "синий")
single_element = (42,)  # запятая обязательна для одного элемента
```

Словари (dictionary) — коллекции пар ключ-значение Позволяют быстро находить значения по ключам.

```
person = {"имя": "Анна", "возраст": 28, "город": "Москва"}
empty_dict = {}
nested_dict = {"пользователь": {"имя": "Иван", "роль": "администратор"}}
```

Множества (set) — неупорядоченные коллекции уникальных элементов Автоматически удаляют дубликаты.

```
unique_numbers = {1, 2, 3, 4, 5}
fruits = {"яблоко", "банан", "апельсин"}
empty_set = set()  # пустое множество создается функцией set()
```

Практические примеры работы с переменными

Создание и присваивание значений

Простое присваивание

```
x = 5
name = "Алексей"
is_active = True
```

Множественное присваивание

```
a, b, c = 1, 2, 3
x = y = z = 0
```

Обмен значениями

```
a, b = b, a
```

Определение типа данных

```
x = 42
name = "Python"
is_valid = True
print(type(x))        # <class 'int'>
print(type(name))     # <class 'str'>
print(type(is_valid)) # <class 'bool'>
```

Проверка принадлежности к типу

```
print(isinstance(x, int))     # True
print(isinstance(name, str))  # True
```

Изменение значений переменных

```
counter = 0
print(counter)  # 0
counter = 10
print(counter)  # 10
counter += 5    # Увеличение на 5
print(counter)  # 15
```

Преобразование типов данных (Type Casting)

Python позволяет преобразовывать данные из одного типа в другой с помощью встроенных функций.

Преобразование чисел

```
Число в строку
x = 123
x_str = str(x)      # "123"
Целое число в float
x_float = float(x)  # 123.0
Float в целое число
y = 3.14
y_int = int(y)      # 3 (дробная часть отбрасывается)
Преобразование строк
Строка в число (если возможно)
num_str = "456"
num_int = int(num_str)    # 456
num_float = float(num_str) # 456.0
Строка в список символов
text = "Python"
char_list = list(text)    # ['P', 'y', 't', 'h', 'o', 'n']
```

Обработка ошибок преобразования

```
try:
invalid_str = "abc123"
number = int(invalid_str)  # Вызовет ValueError
except ValueError:
print("Невозможно преобразовать строку в число")
```

Полезные советы и рекомендации

Проверка содержимого переменной

```
#Проверка на пустое значение
text = ""
if not text:
print("Строка пуста")

#Проверка на None
value = None
if value is None:
print("Значение не определено")

#Работа с глобальными и локальными переменными
global_var = "Глобальная переменная"
def example_function():
    local_var = "Локальная переменная"
    global global_var
    global_var = "Измененная глобальная переменная"
    print(local_var)   # Доступна только внутри функции
    print(global_var)  # Доступна везде

#Константы в Python
#Константы обычно записываются заглавными буквами
PI = 3.14159
MAX_SIZE = 100
DATABASE_URL = "localhost:5432"
```

Заключение

Переменные и типы данных — это фундаментальные концепции Python, которые необходимо понимать для эффективного программирования. Правильное использование переменных и понимание различных типов данных поможет вам создавать более читаемый и эффективный код. Помните о правилах именования переменных, осваивайте работу с различными типами данных и всегда проверяйте корректность преобразований типов в вашем коде.