Лекция 6 - Чтение и вывод данных в Python: основные методы и примеры использования

Ввод и вывод данных (I/O) являются фундаментальными операциями в программировании, позволяющими программам взаимодействовать с пользователем и внешними источниками информации.

Вывод данных в Python: функция print()

Функция print() является основным инструментом для отображения информации в Python. Она предоставляет множество возможностей для форматированного вывода данных на консоль или в файлы.

Основы работы с print()

Простейшее использование функции print() заключается в выводе текста или значений переменных:

```
print("Hello, world!")  # Выводит текстовую строку
x = 10
print("Значение x равно:", x)  # Выводит текст и значение переменной
```

Форматирование вывода в Python

Python предлагает несколько способов форматирования строк для красивого вывода данных:

```
name = "Анна"
age = 25
salary = 50000.50
```

Метод .format()

```
print("Имя: {}, Возраст: {}, Зарплата: {:.2f}".format(name, age, salary))
```

F-строки (рекомендуемый способ)

```
print(f"Имя: {name}, Возраст: {age}, Зарплата: {salary:.2f}")
```

Старый способ форматирования

```
print("Имя: %s, Возраст: %d, Зарплата: %.2f" % (name, age, salary))
Параметры функции print()
Функция print() имеет несколько полезных параметров:
```

Параметр sep - определяет разделитель между элементами:

```
print("яблоко", "банан", "апельсин", sep=", ")  # Выводит: яблоко, банан, апельсин
print("2024", "01", "15", sep="-")  # Выводит: 2024-01-15
```

Параметр end - определяет символ окончания строки:

```
print("Загрузка", end="")
for i in range(3):
    print(".", end="")
print(" Готово!")  # Выводит: Загрузка... Готово!
```

Параметр file - позволяет направить вывод в файл:

```
with open("log.txt", "w", encoding="utf-8") as file:
    print("Информация сохранена в файл", file=file)
```

Ввод данных в Python: функция input()

Функция input() позволяет получать данные от пользователя в интерактивном режиме. Она всегда возвращает строку, которую при необходимости нужно преобразовывать в другие типы данных.

Основы работы с input()

```
# Простой ввод без приглашения
name = input()
print(f"Привет, {name}!")

# Ввод с приглашением
age = input("Введите ваш возраст: ")
print(f"Вам {age} лет")
```

Преобразование типов данных

Поскольку input() всегда возвращает строку, необходимо преобразовывать данные в нужный тип:

```
# Преобразование в целое число
age = int(input("Введите ваш возраст: "))
# Преобразование в число с плавающей точкой
height = float(input("Введите ваш рост в метрах: "))
# Обработка списка значений
numbers = input("Введите числа через пробел: ").split()
numbers = [int(num) for num in numbers]
```

Обработка ошибок ввода

Важно предусматривать обработку некорректного ввода:

```
while True:
    try:
        age = int(input("Введите ваш возраст: "))
        if age < 0:
            print("Возраст не может быть отрицательным")
            continue
        break
    except ValueError:
        print("Пожалуйста, введите корректное число")
print(f"Ваш возраст: {age}")
```

Валидация пользовательского ввода

```
def get_valid_email():
    while True:
        email = input("Введите email: ")
        if "@" in email and "." in email:
            return email
        print("Некорректный формат email")
email = get_valid_email()
print(f"Ваш email: {email}")
```

Продвинутые техники ввода-вывода

Форматирование чисел и дат

```
from datetime import datetime
```

Форматирование чисел

```
price = 1234.5678
print(f"Цена: {price:.2f} руб.")  # Цена: 1234.57 руб.
print(f"Цена: {price:,.2f} руб.")  # Цена: 1,234.57 руб.
```

Форматирование дат

```
now = datetime.now()
print(f"Текущая дата: {now:%d.%m.%Y}")
print(f"Текущее время: {now:%H:%M:%S}")
```

Создание интерактивных программ

```
def calculator():
    print("Простой калькулятор")
    print("Доступные операции: +, -, *, /")
    while True:
        try:
            num1 = float(input("Введите первое число: "))
            operator = input("Введите операцию (+, -, *, /): ")
            num2 = float(input("Введите второе число: "))
            if operator == "+":
                result = num1 + num2
            elif operator == "-":
                result = num1 - num2
            elif operator == "*":
                result = num1 * num2
            elif operator == "/":
                if num2 == 0:
                    print("Деление на ноль невозможно!")
                    continue
                result = num1 / num2
            else:
                print("Неизвестная операция")
                continue
            print(f"Результат: {result}")
            if input("Продолжить? (y/n): ").lower() != 'y':
                break
        except ValueError:
            print("Ошибка: введите корректное число")
calculator()
```

Лучшие практики работы с I/O в Python

1. Валидируйте пользовательский ввод

```
def get_positive_number(prompt):
    while True:
        try:
            num = float(input(prompt))
            if num > 0:
                return num
            print("Число должно быть положительным")
        except ValueError:
            print("Введите корректное число")
```

Заключение

Ввод и вывод данных в Python предоставляют программистам мощные инструменты для создания интерактивных приложений и обработки информации. Понимание функций print() и input()

Основные принципы работы с I/O в Python:

Используйте f-строки для форматирования вывода

Всегда обрабатывайте исключения при работе с файлами

Валидируйте пользовательский ввод