Конструкции языка программирования (Python)

Алгоритмы и среда программирования.

Переменные в Python

- Переменная — это контейнер для хранения данных.
- В Python не требуется объявлять тип переменной заранее.

Пример:

```
x = 10  # Целое число
name = "Alice"  # Строка
pi = 3.14  # Число с плавающей точкой
is_active = True  # Логическое значение
```

Переменные в Python (2)

- Использование функции **type()** для определения типа данных.

Пример:

```
x = 10
print(type(x))  # Выведет: <class 'int'>
```

- **Динамическая типизация:** можно изменять тип данных переменной

Пример:

```
x = 10
x = "Hello"  # Теперь x хранит строку
```

Отступы в Python

Отступ

- В Python отступы (пробелы или табуляция) определяют блоки кода.
- Ошибки, связанные с отступами (IndentationError).

Пример

```
if True:
    print("Отступ важен!")
```

Условные операторы (if-elif-else)

if-elif-else

Условные операторы используются для принятия решений в коде.

Пример

```
x = int(input("Введите число: "))
if x > 0:
    print("Положительное число")
elif x < 0:
    print("Отрицательное число")
else:
    print("Ноль")
```

Условные операторы (if-elif-else) Синтаксис

```
if условие:
    # выполняется, если условие истинно
elif другое_условие:
    # выполняется, если первое условие ложно, а второе истинно
else:
    # выполняется, если все условия ложны
```

Условные операторы (if-elif-else) Вложенные условия

- Вложенные условия позволяют создавать более сложные проверки

```
x = int(input("Введите число: "))
if x > 0:
    if x % 2 == 0:
        print("Число положительное и четное")
    else:
        print("Число положительное и нечетное")
elif x < 0:
    print("Число отрицательное")
else:
    print("Число равно нулю")
```

Циклы (for, while)

- **Циклы** позволяют выполнять блок кода многократно.
- В Python существуют два основных типа циклов: **for** и **while**.

Цикл for

- Используется, когда заранее известное количество повторений.
- Часто применяется с функцией **range()**

- Пример:

```
for i in range(5):
    print("Итерация:", i)
```

- Пример перебора элементов списка:

```
fruits = ["яблоко", "банан", "вишня"]
for fruit in fruits:
    print(fruit)
```

Цикл while

- Используется, когда количество итераций заранее неизвестно.
- Работает, пока выполняется условие.

- Пример:

```
n = 5
while n > 0:
    print("Число:", n)
    n -= 1
```

Операторы **break** и **continue**

- **Операторы break и continue**
- **break** – завершает выполнение цикла досрочно.
- **continue** – пропускает текущую итерацию и переходит к следующей.

- Пример:

```
for i in range(10):
    if i == 5:
        break  # Прерывание цикла, когда i равно 5
    print(i)
```

- Пример перебора элементов списка:

```
for i in range(10):
    if i % 2 == 0:
        continue  # Пропускаем четные числа
    print(i)
```

https://github.com/agorbatniov/inf10

Функции

- Создание функций, аргументы и возвращаемые значения.
- Локальные и глобальные переменные.

Пример

```
def add(a, b):
    return a + b
print(add(3, 5))
```

Рекурсия

Пример

- Что такое рекурсия?
- Пример рекурсивной функции (факториал).

```
def factorial(n):
    if n == 1:
        return 1
    return n * factorial(n - 1)
print(factorial(5))
```

Практическое задание

- **Задача 1. Четные числа**

Напишите программу, которая выводит четные числа от 1 до 20 с использованием цикла **for**.

- **Задача 2. Факториал с использованием рекурсии**

Реализуйте программу, которая вычисляет факториал числа, введенного пользователем.

Задача 1. Четные числа

- Напишите программу, которая выводит четные числа от 1 до 20 с использованием цикла **for**.

Решение:

```
for i in range(1, 21):
    if i % 2 == 0:
        print(i)
```

Задача 2. Факториал с использованием рекурсии

Реализуйте программу, которая вычисляет факториал числа, введенного пользователем.

- **Решение:**

```
def factorial(n):
    if n == 0 or n == 1:
        return 1
    return n * factorial(n - 1)

num = int(input("Введите число: "))
print("Факториал:", factorial(num))
```

Конец