Алгоритмы и структуры данных на Python. Интерактивный курс

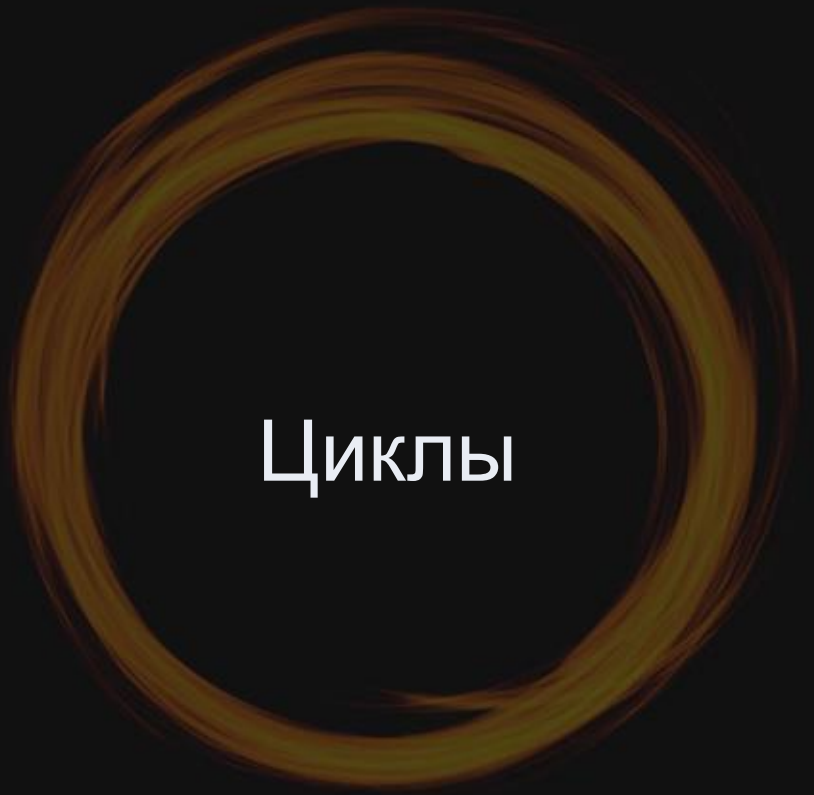
Урок 2

Циклы. Рекурсия. Функции

Циклы. Рекурсивный перебор. Алгоритм Евклида. Решето Эратосфена

План

- Что такое циклический алгоритм
- Виды циклических алгоритмов
- Графическое представление циклического алгоритма
- Примеры циклов на Python

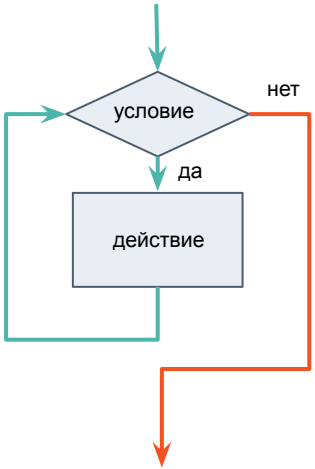


Циклы

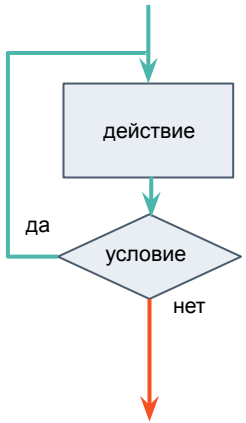
Понятие цикла

- Цикл с предусловием
- Цикл с постусловием
- Арифметический цикл (цикл с параметром)

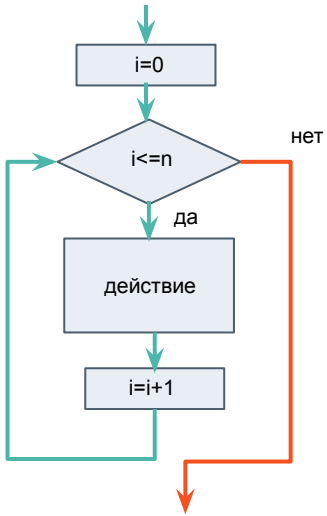
Алгоритмическое представление цикла с предусловием



Алгоритмическое представление цикла с постусловием



Алгоритмическое представление цикла с параметром



Итоги:

Теория

- Циклический алгоритм
- Виды циклических алгоритмов

Практика

- Блок-схемы циклических алгоритмов
- Примеры циклов на Python

План

- Что такое рекурсия
- Как написать рекурсию (шаги рекурсии)
- Графическое представление рекурсии
- Примеры рекурсии на Python

Рекурсия

Определение части функции (метода) через саму себя.

Запомните!!!

Любой алгоритм, реализованный с помощью рекурсивного перебора, может быть представлен в итерационном виде и наоборот.

Основные шаги рекурсивной функции

- **Первый шаг.** Необходимое условие для остановки (базовый случай)
- **Второй шаг.** Необходимое условие для продолжения или шаг рекурсии

Закрепляем изученное

- **Задача.** Даны два целых числа: A и B.

  Необходимо вывести все числа от A до B включительно, в порядке возрастания, если A < B, или в порядке убывания, если A > B.

Итоги:

Теория

- Рекурсия
- Шаги рекурсии

Практика

- Блок-схема рекурсивного алгоритма
- Примеры рекурсии на Python

План

- **Задача.** Реализация функции Аккермана с помощью рекурсии

Функции Аккермана

$$A(m, n) = \begin{cases} n + 1, & m = 0; \\ A(m - 1, 1), & m > 0, n = 0 \\ A(m - 1, A(m, n - 1)), & m > 0, n > 0 \end{cases}$$

Итоги:

Практика

- Блок-схема рекурсивного алгоритма функция Аккермана
- Реализация функции Аккермана на Python

План

- Алгоритм Евклида.
- **Задача.** Найти наибольший общий делитель (НОД, greatest common divisor, gcd) пары чисел.

Пример: для чисел 54 и 24 НОД равен 6.

Алгоритм Евклида

- **Вариант 1.** Простейший циклический алгоритм основанный на вычитании.
- **Вариант 2.** Рекурсивный алгоритм основанный на нахождении остатка от деления.
- **Вариант 3.** Циклический алгоритм основанный на нахождении остатка от деления.

Итоги:

Практика

- Блок-схемы алгоритма Евклида
- Реализация алгоритма Евклида на Python

План

- Решето Эратосфена.
- **Задача.** Найти простые числа до заданного числа N.

  Пример: Если N == 15, то вернём [2, 3, 5, 7, 11, 13].

Решето Эратосфена

~~1~~	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40

Решето Эратосфена

~~1~~	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40

Решето Эратосфена

~~1~~	2	3	~~4~~	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40

Решето Эратосфена

~~1~~	2	3	~~4~~	5	~~6~~	7	~~8~~	9	~~10~~
11	~~12~~	13	~~14~~	15	~~16~~	17	~~18~~	19	~~20~~
21	~~22~~	23	~~24~~	25	~~26~~	27	~~28~~	29	~~30~~
31	~~32~~	33	~~34~~	35	~~36~~	37	~~38~~	39	~~40~~

Решето Эратосфена

~~1~~	2	3	~~4~~	5	~~6~~	7	~~8~~	~~9~~	~~10~~
11	~~12~~	13	~~14~~	~~15~~	~~16~~	17	~~18~~	19	~~20~~
~~21~~	~~22~~	23	~~24~~	~~25~~	~~26~~	~~27~~	~~28~~	29	~~30~~
31	~~32~~	~~33~~	~~34~~	35	~~36~~	37	~~38~~	~~39~~	~~40~~

Решето Эратосфена

~~1~~	2	3	~~4~~	5	~~6~~	7	~~8~~	~~9~~	~~10~~
11	~~12~~	13	~~14~~	~~15~~	~~16~~	17	~~18~~	19	~~20~~
~~21~~	~~22~~	23	~~24~~	~~25~~	~~26~~	~~27~~	~~28~~	29	~~30~~
31	~~32~~	~~33~~	~~34~~	~~35~~	~~36~~	37	~~38~~	~~39~~	~~40~~

Решето Эратосфена

~~1~~	2	3	~~4~~	5	~~6~~	7	~~8~~	~~9~~	~~10~~
11	~~12~~	13	~~14~~	~~15~~	~~16~~	17	~~18~~	19	~~20~~
~~21~~	~~22~~	23	~~24~~	~~25~~	~~26~~	~~27~~	~~28~~	29	~~30~~
31	~~32~~	~~33~~	~~34~~	~~35~~	~~36~~	37	~~38~~	~~39~~	~~40~~

Итоги:

Практика

- Блок-схема Решета Эратосфена
- Реализация алгоритма Решето Эратосфена на Python

План

- Практическое занятие
- Домашнее задание

Закрепляем изученное

- **Задача.** Функция перевода десятичного числа в двоичный формат

Домашние задания

- Для каждого упражнения составьте графическое представление алгоритма и напишите программный код
- Сохраняйте каждую задачу (код на Python) в отдельный файл
- В начале файла вставьте текст задачи в виде комментария
- Графическое представление (блок-схему) присылайте в удобном для просмотра виде (jpg, pdf и т.п.)

Домашнее задание

1. Написать программу, которая будет складывать, вычитать, умножать или делить два числа. Числа и знак операции вводятся пользователем. После выполнения вычисления программа должна не завершаться, а запрашивать новые данные для вычислений. Завершение программы должно выполняться при вводе символа '0' в качестве знака операции. Если пользователь вводит неверный знак (не '0', '+', '-', '*', '/'), то программа сообщает ему об ошибке и снова запрашивает знак операции. Также пользователю нужно сообщать о невозможности деления на ноль, если он ввел 0 в качестве делителя.

Домашнее задание

2. Посчитать четные и нечетные цифры введенного натурального числа. Например, если введено число 34560, то у него 3 четные цифры (4, 6 и 0) и 2 нечетные (3 и 5).

Домашнее задание

3. Сформировать из введенного числа обратное по порядку входящих в него цифр и вывести на экран. Например, если введено число 3486, то надо вывести 6843.

Домашнее задание

4. Найти сумму n элементов следующего ряда чисел:

   1, -0.5, 0.25, -0.125, …

   Количество элементов (n) вводится с клавиатуры.

Домашнее задание

5. Вывести на экран коды и символы таблицы ASCII, начиная с символа под номером 32 и заканчивая 127-м включительно. Вывод выполнить в табличной форме: по десять пар «код-символ» в каждой строке.

Домашнее задание

6. Написать программу, где генерируется случайное целое число от 0 до 100. Пользователь должен его отгадать максимум за 10 попыток. После каждой неудачи должно сообщаться, больше или меньше загаданного то число, что ввел пользователь. Если за 10 попыток число не отгадано – вывести его.

Домашнее задание

7. Написать программу, доказывающую или проверяющую, что для множества натуральных чисел выполняется равенство:

   $$1 + 2 + \ldots + n = n \times (n + 1) / 2,$$

   где n – любое натуральное число.

Домашнее задание

8. Посчитать, сколько раз встречается определенная цифра в введенной последовательности чисел. Количество вводимых чисел и цифра, которую необходимо посчитать, задаются вводом с клавиатуры.

Пример: 4
8
15469
5648
165489
6161

Домашнее задание

9. Среди натуральных чисел, которые были введены, найти наибольшее по сумме цифр. Вывести на экран это число и сумму его цифр.

Итоги:

Теория

- Требования к домашнему заданию и само задание

Практика

- Блок-схема
- Код на Python

План

- Разбор домашнего задания