АКАДЕМИЯ АЙТИ

День 4

academyit.ru

Модели разработки программ. Структурное программирование. Базовые принципы:
* блочная структура кода – блоки и подпрограммы.
* типовые структуры управления – последовательность, ветвление, цикл

Турашова Анна Николаевна
Преподаватель
anna1turashova@gmail.com
Telegram: @anna1tur

Логический тип данных

- Выражения логического типа в Python принимают одно из двух значений True (истина) и False (ложь)
- Логический тип называется bool в честь Джорджа Буля
- Условия ==, !=, <, >, <=, >= вычисляют значение логического типа
- Для логического типа можно использовать специальные логические операции

Логическое умножение (and, и)

a	b	a and b
False	False	False
False	True	False
True	False	False
True	True	True

- Логическое выражение a and b **истинно**, только если **оба** значения a и b **истинны**
- В общем случае значение выражения с оператором and истинно, если истинны **все** объединенные им условия

Пример на логическое умножение

Напишите программу, которая получает номер месяца и выводит соответствующее ему время года или сообщение об ошибке.

Пример:

```
5
Весна
```

Пример:

```
15
Неверный номер месяца
```

```
m = int(input())
if m >= 6 and m <= 8:
    print('Лето')
```

Логическое сложение (or, или)

a	b	a or b
False	False	False
False	True	True
True	False	True
True	True	True

- Логическое выражение a or b **истинно**, если **хотя бы одно** значение a и b **истинно**
- В общем случае значение выражения с оператором or истинно, если истинно **хотя бы одно** условие

Пример на логическое сложение

```
m = int(input())
if m == 12 or m <= 2:
    print('Зима')
```

Логическое отрицание (not, не)

a	not a
False	True
True	False

- Логическое выражение not a **истинно**, если a ложно и наоборот

Сложные условия

Задача: набор сотрудников в возрасте **25-40 лет** (включительно)

сложное условие

```
if v >= 25 and v <= 40 :
  print("подходит")
else:
  print("не подходит")
```

Приоритет :

1) отношения (**<**, **>**, **<=**, **>=**, **==**, **!=**)
2) **not** («НЕ»)
3) **and** («И»)
4) **or** («ИЛИ»)

Примеры задач на сложные условия

Задача №296. Количество равных из трех

Входные данные

Даны три целых числа, записанных в отдельных строках. Определите, сколько среди них совпадающих.

Выходные данные

Программа должна вывести одно из чисел: 3 (если все совпадают), 2 (если два совпадают) или 0 (если все числа различны).

Задача №258. Шоколадка

Требуется определить, можно ли от шоколадки размером $n \times m$ долек отломить k долек, если разрешается сделать один разлом по прямой между дольками (то есть разломить шоколадку на два прямоугольника).

Входные данные

Вводятся 3 числа: n, m и k; k не равно $n \times m$. Гарантируется, что количество долек в шоколадке не превосходит 30000.

Выходные данные

Программа должна вывести слово YES, если возможно отломить указанное число долек, в противном случае вывести слово NO.

Примеры задач на сложные условия

Задача №303. Коровы

По данному числу *n* закончите фразу "На лугу пасется..." одним из возможных продолжений: "*n* коров", "*n* корова", "*n* коровы", правильно склоняя слово "корова".

Входные данные

Дано число *n* (*n*<100).

Выходные данные

Программа должна вывести введенное число *n* и одно из слов (на латинице): `korov`, `korova` или `korovy`, например, `1 korova`, `2 korovy`, `5 korov`. Между числом и словом должен стоять ровно один пробел.

Задача №256. Ферзь

Требуется определить, бьет ли ферзь, стоящий на клетке с указанными координатами (номер строки и номер столбца), фигуру, стоящую на другой указанной клетке.

Входные данные

Вводятся четыре числа: координаты ферзя и координаты другой фигуры. Координаты - целые числа в интервале от 1 до 8.

Выходные данные

Требуется вывести слово YES, если ферзь может побить фигуру за 1 ход, в противном случае вывести слово NO

Цикл с параметром for

Арифметический цикл

for переменная_цикла in последовательность:
блок кода (тело цикла)

- Цикл for применяется, когда известно количество повторений (итераций)
- Для переменных цикла используют обычно i, j, k

Повторение n раз

for переменная_цикла in range(n):
блок кода (тело цикла)

```
for i in range(10):
    print('Привет')
```

- Переменная изменяется от 0 до n-1

```
for i in range(5):
    print(i + 1)
```

Повторение от a до b (не включая b)

for переменная_цикла in range(a, b):
**   блок кода (тело цикла)**

```
for i in range(1, 11):
    print(i)
```

- Переменная изменяется от a до b -1
- Если первый параметр больше либо равен второго, то функция range() генерирует пустую последовательность

Повторение от a до b с шагом c

for переменная_цикла in range(a, b, c):
**   блок кода (тело цикла)**

```
for i in range(2, 11, 2):
    print(i)

for i in range(5, 0, -1):
    print(i)
```

- Переменная изменяется от a до b -1 с шагом c
- Шаг может быть отрицательным, тогда последовательность будет убывающая

Примеры задач

Задача №333. Четные числа

Входные данные
Вводятся целые числа *a* и *b*. Гарантируется, что *a* не превосходит *b*

Выходные данные
Выведите (через пробел) все четные числа от *a* до *b* (включительно).

Задача №343. Сумма чисел

Вычислите сумму данных *N* натуральных чисел.

Входные данные
Вводится число *N*, а затем *N* чисел, сумму которых необходимо вычислить.

Выходные данные
Выведите единственное число - сумму введенных чисел.

Задача №351. Факториал

Вычислите *N*! ("эн-факториал") – произведение всех натуральных чисел от 1 до *N* ($N!=1\cdot2\cdot3\cdot\ldots\cdot N$).

Входные данные
Вводится единственное число *N* – натуральное, не превосходит 12.

Выходные данные
Выведите полученное значение *N*!

Вычисление суммы

Задача №343. Сумма чисел

Вычислите сумму данных *N* натуральных чисел.

Входные данные

Вводится число *N*, а затем *N* чисел, сумму которых необходимо вычислить.

Выходные данные

Выведите единственное число - сумму введенных чисел.

```
n = int(input())
total = 0  # сумма
for i in range(n):  # повторить n раз
    x = int(input())
    total += x  # total = total + x
print(total)
```

Вычисление произведения

Задача №351. Факториал

Вычислите *N*! ("эн-факториал") – произведение всех натуральных чисел от 1 до *N* (*N*!=1·2·3·...· *N*).

Входные данные

Вводится единственное число *N* – натуральное, не превосходит 12.

Выходные данные

Выведите полученное значение *N*!

```
n = int(input())
f = 1  # факториал
for i in range(1, n + 1):
    f *= i  # f = f * i
print(f)
```

Вычисление количества

Задача №346. Подсчет чисел

Подсчитайте, сколько среди данных *N* чисел нулей, положительных чисел, отрицательных чисел.

Входные данные

Вводится число *N*, а затем *N* целых чисел.

Выходные данные

Необходимо вывести сначала число нулей, затем число положительных и отрицательных чисел.

```
n = int(input())
count0 = 0
countp = 0
countn = 0
for i in range(n):
    x = int(input())
    if x == 0:
        count0 += 1
    elif x > 0:
        countp += 1
    else:
        countn += 1
print(count0, countp, countn)
```

Использование флагов

- Флаг – логическая переменная, которой присваивается значение True при наступлении некоторого события

Задача №347. Ноль или не ноль

Проверьте, есть ли среди данных *N* чисел нули.

Входные данные

Вводится число *N*, а затем *N* чисел.

Выходные данные

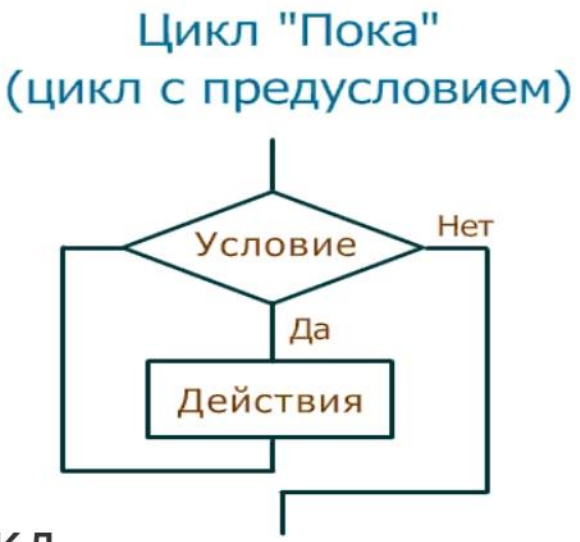
Выведите YES, если среди введенных чисел есть хотя бы один нуль, или NO в противном случае.

```
n = int(input())
flag = False  # был ли ноль?
for i in range(n):
    x = int(input())
    if x == 0:
        flag = True  # ноль был!
if flag:  # flag == True
    print("YES")
else:
    print("NO")
```

Цикл while (с предусловием)

- Цикл while (“пока”) позволяет выполнить одну и ту же последовательность действий, пока проверяемое условие истинно.

```
while условие:
    блок инструкций
```



- условие проверяется при входе в цикл
- как только условие становится ложным, работа цикла заканчивается
- если условие **ложно в самом начале**, цикл не выполняется **ни разу**

Пример

Вывести числа от 1 до 10.

```
i = 1
while i <= 10:
    print(i)
    i += 1
```

Работа с цифрами целого числа

Задача Сумма цифр

Выделить последнюю цифру числа в переменной d:

```
d = n % 10
```

$123 \to 3$

Отбросить последнюю цифру числа в переменной n:

```
n = n // 10
```

$123 \to 12$

Добавить цифру в сумму:

```
sum += d
```

$sum = 0 \to 3$

$d = 3$

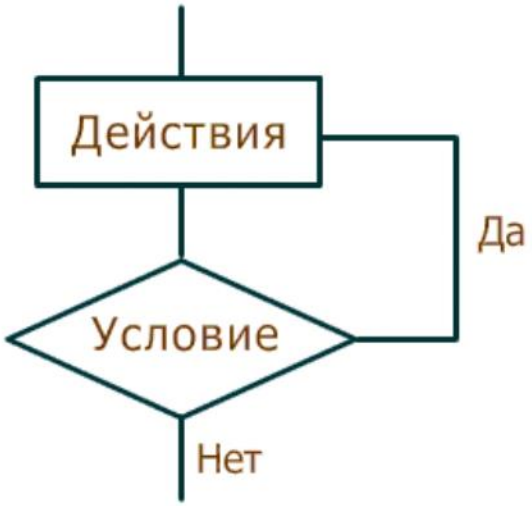
Решение Сумма цифр

```
n = int(input())
sum = 0
while n > 0:
    d = n % 10
    n //= 10
    sum += d
print(sum)
```

Цикл с постусловием

- Цикл с постусловием можно реализовать с помощью while, если вынести действия до цикла

Цикл с постусловием




```
блок кода
while условие:
    блок кода
```

- условие проверяется после первого выполнения блока кода

Обработка последовательностей

Задача. На вход программы поступает поток данных — последовательность целых чисел, которая **заканчивается нулём**. Требуется найти сумму элементов этой последовательности.

```
while x!=0:
   # добавить x к сумме
   # x = следующее число
```

Откуда возьмётся `x` в первый раз?

Решение

```
sum = 0
x = int(input()) # первое число
while x!=0:
   sum += x
   x = int(input()) # ввести следующее
print("Сумма ", sum)
```

Управление циклом

break (“прервать”) - выходит из цикла

continue (“продолжить”) – пропускает остаток блока и переходит на заголовок цикла

else:

    блок кода

- выполняется, если цикл не был прерван

Простое число

Ввести число. Вывести простое оно или составное.

```
n = int(input())
d = 2
while d * d <= n:
    if n % d == 0:
        print('составное')
        break
    d += 1
else:
    print('простое')
```

Домашние задания:

Задание 1

Напишите программу, которая имитирует проверку пароля, придуманного пользователем.

В переменной password хранится правильный пароль.

Пользователь вводит слово. Если оно совпадает с паролем, программа отвечает «вы вошли в аккаунт», если не совпадает – пользователь вновь вводит слово.

Задание 2

Добавьте для предыдущей задачи количество попыток ввода пароля.

Если пользователь превысит это количество, то программа сообщит об этом и закончит работу.

Задание 3

Том хотел написать программу, которая будет удалять из строки все заглавные буквы.

Для написания такой программы он воспользовался методом .replace() , который принимает два аргумента:
1 – это символ, который нужно найти в строке,
2 – это символ, на которой нужно заменить найденный символ.

Но его программа работает не корректно:

```
letters = 'ВДяЫгВЫеЯСнДШКАиЩЙФй'

for letter in letters:
    if letter.upper() = letters:
        letters.replace(letter, '')

print(letters)
```

Исправьте программу. Или напишите свой алгоритм решения.

Задание 4

Дан список list_num, который содержит любые числа.
Создайте новый пустой список list_new.
Напишите программу, которая заполнит список list_new только такими элементами из списка list_num, которые больше пяти по модулю.

Задание 5

Пользователь вводит число n.
По данному числу необходимо найти наибольшую целую степень двойки, не превосходящую n.

Например:
n = 9, 2**3 < n, 2**4 > n, соответственно степень 3.
n = 1963, 2**10 < n, 2**11 > n, соответственно степень 10.

АПЛАНА
ГРУППА КОМПАНИЙ

Входит в ГК Аплана

АКАДЕМИЯ АЙТИ

Основана в 1995 г.

E-learning и очное обучение

Направления обучения:

- Информационные технологии
- Информационная безопасность
- ИТ-менеджмент и управление проектами
- Разработка и тестирование ПО
- Гос. и муниципальное управление

Филиалы:

Санкт-Петербург, Казань, Уфа, Челябинск, Хабаровск, Красноярск, Тюмень, Нижний Новгород, Краснодар, Волгоград, Ростов-на-Дону

Головной офис в Москве

Разработка программного обеспечения и информационных систем

Ежегодные награды Microsoft, Huawei, Cisco и другие

Программы по импортозамещению

Ресурсы более 400 высококлассных экспертов и преподавателей

Сеть региональных учебных центров по всей России

Крупные заказчики

СБЕР

РЖД

Билайн

TATNEFT

НОРНИКЕЛЬ

100+
сотрудников

АКАДЕМИЯ АЙТИ

Спасибо за внимание!

Центральный офис:
Москва, Варшавское шоссе 47, корп. 4, 7 этаж
Тел: +7 (495) 150-96-00
academy@it.ru
academyit.ru