АЛГОРИТМИЗАЦИЯ И ПРОГРАММИРОВАНИЕ

и.о.доцент Муханова А.А.

Лекция 4. Алгоритмы решения задач с использованием операторов цикла. Структура операторов цикла.

План лекции

Инструкция for
Инструкция цикла while
Вложенные циклы
Оператор break.
Оператор continue

- *Циклы* — это инструкции, выполняющие одну и ту же последовательность действий, пока действует заданное условие.

Циклом называется фрагмент алгоритма или программы, который может повторяться несколько раз (в том числе и нуль раз). Каждая циклическая конструкция начинается заголовком цикла и заканчивается конечным оператором. Между ними располагаются операторы, называемые «телом цикла».

Количество повторений выполнения команд (операторов), составляющих тело цикла, определяется условием окончания цикла. Условием окончания может быть достижение некоторого значения специальной переменной, называемой параметром цикла (переменной цикла), или выполнение (прекращение выполнения) некоторого условия.

Для организации циклов с параметром в языках программирования используется составной оператор FOR («для»), а в циклах с условием чаще всего используется составной оператор WHILE ("пока логическое выражение возвращает истину, выполнять определенные операции").

Инструкция for

Она применяется в тех случаях, когда в программе какие-то действия (инструкции) повторяются и при этом некоторая величина меняется с постоянным шагом.

Пример 1. Для вывода «столбиком» всех целых чисел от 10 до 20 без использования инструкции for в программе потребовалось бы записать:

```
print(10)
print(11)
...
print(20)
```

Видно, что есть повторяющиеся действия (вывод на экран числа) и что при этом выводимое число меняется с шагом, равным 1. Можно применить инструкцию for.

Инструкция for

Общий вид этой инструкции:

for <параметр инструкции> in <набор значений>:

<тело инструкции> #Записывается с отступом

- где <тело инструкции>– действия, повторяющиеся при работе инструкции (это могут быть любые инструкции Python);
- <параметр инструкции> – имя величины, меняющейся при повторении действий;
- <набор значений>– набор значений параметра инструкции, для которых проводится повторение.

Пример:

```
>>> num=[0.8, 7.0, 6.8, -6]
>>> for i in num:
        print(i, '- number')
0.8 - number
7.0 - number
6.8 - number
-6 - number
>>>
```

Примеры

```
>>> for i in [1, 2, 'hi']:
print(i)
1
2
hi
>>>
```

На самом деле, цикл for работает и для строк!

```
>>> for i in 'hello':
print(i)
h
e
l
l
o
>>>
```

Функция range()

- Достаточно часто при разработке программ необходимо получить последовательность (диапазон) целых чисел

В качестве аргументов функция принимает:
начальное значение диапазона (по умолчанию 0), конечное значение (не включительно) и шаг (по умолчанию 1).

Функция range()

```
>>> for i in range(0, 10, 1):
        print(i, end=' ')
0 1 2 3 4 5 6 7 8 9

>>> for i in range(10):
        print(i, end=' ')
0 1 2 3 4 5 6 7 8 9

>>> for i in range(2, 20, 2):
        print(i, end=' ')
2 4 6 8 10 12 14 16 18
>>>
```

Таким образом, в переменную i на каждом шаге цикла будет записываться значение из диапазона, который создается функцией range().

При желании можно получить диапазон в обратном порядке следования (обратите внимание на аргументы функции range()):

```
>>> for i in range(20, 2, -2):
        print(i, end=' ')
20 18 16 14 12 10 8 6 4
>>>
```

Пример. Функция range()

Теперь с помощью диапазона найдем сумму чисел на интервале от 1 до 100:

```
>>> total=0
>>> for i in range(1, 101):
        total=total+i
>>> total
5050
>>>
```

Переменной i на каждом шаге цикла будет присваиваться значение из диапазона от 1 до 100 (крайнее значение не включаем). В цикле мы накапливаем счетчик. Что это означает?

На первом шаге цикла сначала вычисляется правая часть выражения, т.е. total+i.

Переменная total на первом шаге равна 0 (присвоили ей значение 0 перед началом цикла), переменная i на первом шаге содержит значение 1 (первое значение из диапазона), таким образом, правая часть будет равна значению 1 и это значение присвоится левой части выражения, т.е. переменной total.

На втором шаге total уже будет равна значению 1, i – содержать значение 2, т.е. правая

часть выражения будет равна 3, это значение присвоится снова total и т.д. пока не дойдем до конца диапазона. В итоге в total после выхода из цикла будет содержаться искомая сумма!

Инструкция цикла while

Цикл for используется, если заранее известно, сколько повторений необходимо выполнить (указывается через аргумент функции range() или пока не закончится список/строка).
Если заранее количество повторений цикла неизвестно, то применяется другая конструкция, которая называется циклом **while:**

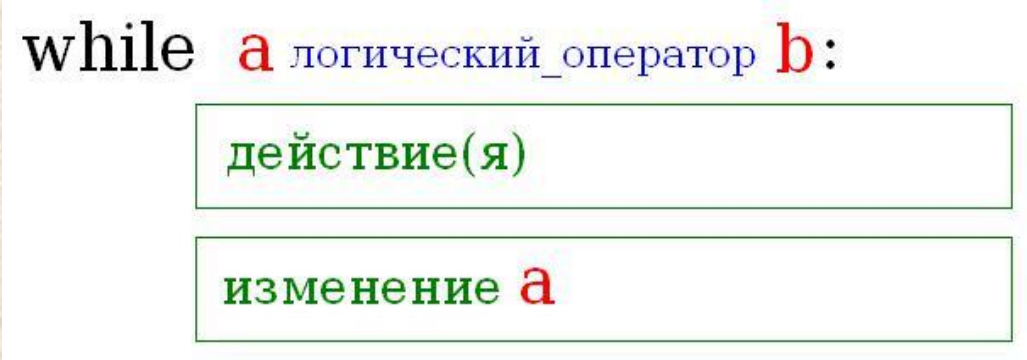
Определим количество кроликов:

```
rabbits = 3
while rabbits > 0:
    print(rabbits)
    rabbits = rabbits – 1
```

В результате выполнения программы:
В примере цикл while выполняется до тех пор, ПОКА число кроликов в условии положительное. На каждом шаге цикла мы переменную rabbits уменьшаем на 1, чтобы не уйти в бесконечный цикл, когда условие всегда будет являться истинным.

Инструкция цикла while

- Универсальным организатором цикла в языке программирования Python (как и во многих других языках) является конструкция **while**. Слово "while" с английского языка переводится как "пока" ("пока логическое выражение возвращает истину, выполнять определенные операции"). Конструкцию **while** на языке Python можно описать следующей схемой:

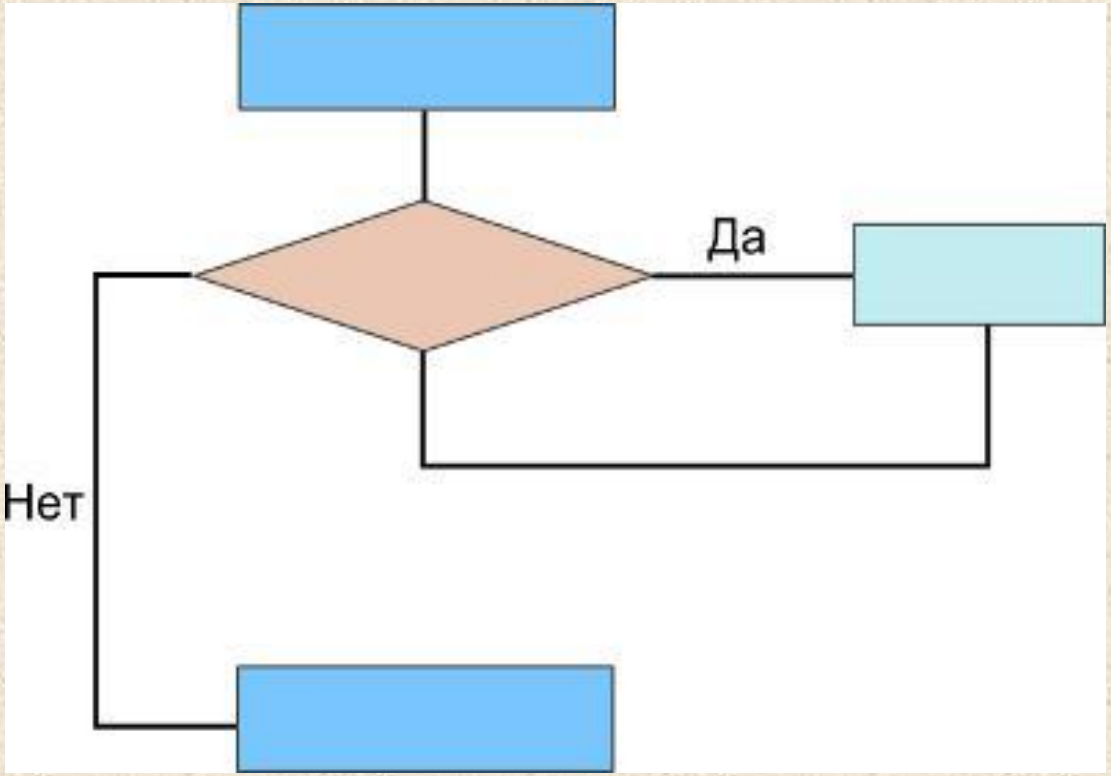


Инструкция цикла while

Эта схема приблизительна, т.к. логическое выражение в заголовке цикла **while** может быть более сложным, а изменяться может переменная (или выражение) b.

Может возникнуть вопрос: "Зачем изменять a или b?". Когда выполнение программного кода доходит до цикла **while**, выполняется логическое выражение в заголовке, и, если было получено **True** (истина), выполняются вложенные выражения. После поток выполнения программы снова возвращается в заголовок цикла **while**, и снова проверяется условие. Если условие никогда не будет ложным, то не будет причин остановки цикла и программа *зациклится*. Чтобы этого не произошло, необходимо предусмотреть возможность выхода из цикла — ложность выражения в заголовке. Таким образом, изменяя значение переменной в теле цикла, можно довести логическое выражение до ложности.

Эту изменяемую переменную, которая используется в заголовке цикла **while**, обычно называют *счетчиком*. Как и всякой переменной ей можно давать произвольные имена, однако очень часто используют буквы i и j.



Простейший цикл на языке программирования Python может выглядеть так:

```
str1 = "+"
i = 0
while i < 10:
    print (str1)
    i = i + 1
```

В последней строчке кода происходит увеличение значения переменной i на единицу, поэтому с каждым оборотом цикла ее значение увеличивается. Когда будет достигнуто число 10, логическое выражение **i < 10** даст ложный результат, выполнение тела цикла будет прекращено, а поток выполнения программы перейдет на команды следующие за всей конструкцией цикла. Результатом выполнения скрипта приведенного выше является вывод на экран десяти знаков + в столбик. Если увеличивать счетчик в теле цикла не на единицу, а на 2, то будет выведено только пять знаков, т.к цикл сделает лишь пять оборотов.

Более сложный пример с использованием цикла:

```
fib1 = 0
fib2 = 1
print (fib1)
print (fib2)
n = 10
i = 0
while i < n:
    fib _sum = fib1 + fib2
    print (fib_sum)
    fib1 = fib2
    fib2 = fib_sum
    i = i + 1
```

Этот пример выводит *числа Фибоначчи* — ряд чисел, в котором каждое последующее число равно сумме двух предыдущих: 0, 1, 1, 2, 3, 5, 8, 13 и т.д. Скрипт выводит двенадцать членов ряда: два (0 и 1) выводятся вне цикла и десять выводятся в результате выполнения цикла.

```
fib1 = 0
fib2 = 1
print (fib1)
print (fib2)
n = 10
i = 0
```

```
while i < n:
    fib _sum = fib1 + fib2
    print (fib_sum)
    fib1 = fib2
    fib2 = fib_sum
    i = i + 1
```

- Как это происходит? Вводятся две переменные (fib1 и fib2), которым присваиваются начальные значения. Присваиваются значения переменной n и счетчику i, между которыми те или иные математические отношения формируют желаемое число витков цикла. Внутри цикла создается переменная fib_sum, которой присваивается сумма двух предыдущих членов ряда, и ее же значение выводится на экран. Далее изменяются значения fib1 и fib2 (первому присваивается второе, а второму - сумма), а также увеличивается значение счетчика.

Пример

Оператор while

Оператор while позволяет многократно выполнять блок команд до тех пор, пока выполняется некоторое условие. Это один из так называемых операторов цикла. Он также может иметь необязательный пункт else.

Пример:

```
number = 23
running = True
while running:
        guess = int(input('Введите целое число : '))
        if guess == number:
        print('Поздравляю, вы угадали.')
        running = False # это останавливает цикл while
        elif guess < number:
        print('Нет, загаданное число немного больше этого')
else:
        print('Нет, загаданное число немного меньше этого.')
        else:
        print('Цикл while закончен.')
        # Здесь можете выполнить всё что вам ещё нужно
print('Завершение.')
```

Вывод:

```
$ python while.py
Введите целое число : 50
Нет, число несколько меньше.
Введите целое число : 22
Нет, число несколько больше.
Введите целое число : 23
Поздравляю, вы угадали.
Цикл while закончен.
Завершение.
```

Вложенные циклы

Циклы можно вкладывать друг в друга.

```
outer = [1, 2, 3, 4]  # внешний цикл
inner = [5, 6, 7, 8]  # вложенный (внутренний) цикл
for i in outer:
for j in inner:
      print ('i=', i, 'j=', j)
```

В примере цикл for сначала продвигается по всем элементам внешнего цикла (фиксируем i=1), затем переходит к вложенному циклу (переменная j) и проходим по всем элементам вложенного списка. Далее возвращаемся к внешнему циклу (фиксируем следующее значение i=2) и снова проходим по всем элементам вложенного списка.

```
>>>
i= 1 j= 5
i= 1 j= 6
i= 1 j= 7
i= 1 j= 8
i= 2 j= 5
i= 2 j= 6
i= 2 j= 7
i= 2 j= 8
i= 3 j= 5
i= 3 j= 6
i= 3 j= 7
i= 3 j= 8
i= 4 j= 5
i= 4 j= 6
i= 4 j= 7
i= 4 j= 8
>>>
```

Оператор break

Оператор break служит для прерывания цикла, т.е. остановки выполнения команд даже если условие выполнения цикла ещё не приняло значения False или последовательность элементов не закончилась. Важно отметить, что если циклы for или while прервать оператором break, соответствующие им блоки else выполняться не будут.

Пример:

```
while True:
    s = input('Введите что-нибудь : ')
    if s == 'выход':
    break
    print('Длина строки: ', len(s))
print('Завершение')
```

Вывод:

```
Введите что-нибудь : Программировать весело.
Длина строки: 23
Введите что-нибудь : Если работа скучна,
Длина строки: 19
Введите что-нибудь : Чтобы придать ей весёлый тон
-Длина строки: 30
Введите что-нибудь : используй Python!
Длина строки: 23
Введите что-нибудь : выход
Завершение
```

Оператор continue

Оператор continue используется для указания Python, что необходимо пропустить все оставшиеся команды в текущем блоке цикла и продолжить со следующей итерации цикла.

Пример:

```
while True:
    s = input('Введите что-нибудь : ')
    if s == 'выход':
    break
    if len(s) < 3:
    print('Слишком мало')
    continue
    print('Введённая строка достаточной длины')
    # Разные другие действия здесь...
```

Вывод:

```
Введите что-нибудь : а
Слишком мало
Введите что-нибудь : 12
Слишком мало
Введите что-нибудь : абв
Введённая строка достаточной длины
Введите что-нибудь : выход
```

Контрольные вопросы

1. Что такое «цикл»?
2. В каких случаях используют инструкцию for? Каков ее общий вид?
3. Что такое «параметр инструкции for»? Что такое «тело инструкции»? Что может быть использовано в качестве параметра инструкции for?
4. Почему инструкцию while называют «циклом с предусловием»?
5. Для чего используется инструкция break?
6. Для чего используется инструкция continue?
7. Для чего используется функция **range()**
8. Можно ли вкладывать циклы друг в друга? Как это работает?

Практическая работа 1

1. Напишите скрипт на языке программирования Python, выводящий ряд чисел Фибоначчи (см. пример выше). Запустите его на выполнение. Затем измените код так, чтобы выводился ряд чисел Фибоначчи, начиная с пятого члена ряда и заканчивая двадцатым.

2. Напишите цикл, выводящий ряд четных чисел от 0 до 20. Затем, каждое третье число в ряде от -1 до -21.

3. Самостоятельно придумайте программу на Python, в которой бы использовался цикл **while**.

Практическая работа 2

1. Найдите все значения функции y (x) = x степень 2+ 3 на интервале от 10 до 30 с шагом 2.
2. Дано число, введенное с клавиатуры. Определите сумму квадратов нечетных цифр в числе
3. Напишите программу-игру. Компьютер загадывает случайное число, пользователь пытается его угадать. Пользователь вводит число до тех пор, пока не угадает или не введет слово «Выход». Компьютер сравнивает число с введенным и сообщает пользователю больше оно или меньше загаданного.