Лабораторная работа №6
Цикл while

1 Цель работы

Изучить цикл while и научиться применять полученные знания на практике.

2 Краткая теория

2.1 Цикл while

Цикл while («пока») позволяет выполнить одну и ту же последовательность действий, пока проверяемое условие истинно. Условие записывается до тела цикла и проверяется до выполнения тела цикла. Как правило, цикл while используется, когда невозможно определить точное значение количества проходов исполнения цикла.

Синтаксис цикла while в простейшем случае выглядит так:

```
while условие:
    блок инструкций
```

При выполнении цикла while сначала проверяется условие. Если оно ложно, то выполнение цикла прекращается и управление передается на следующую инструкцию после тела цикла while. Если условие истинно, то выполняется инструкция, после чего условие проверяется снова и снова выполняется инструкция. Так продолжается до тех пор, пока условие будет истинно. Как только условие станет ложно, работа цикла завершится и управление передастся следующей инструкции после цикла.

Например, следующий фрагмент программы напечатает на экран квадраты всех целых чисел от 1 до 10. Видно, что цикл while может заменять цикл for … in range(…):

```
i = 1
while i <= 10:
    print(i ** 2)
    i += 1
```

В этом примере переменная i внутри цикла изменяется от 1 до 10. Такая переменная, значение которой меняется с каждым новым проходом цикла, называется счетчиком. Заметим, что после выполнения этого фрагмента значение переменной i будет равно 11, поскольку именно при i == 11 условие i <= 10 впервые перестанет выполняться.

Вот еще один пример использования цикла while для определения количества цифр натурального числа n:

```
n = int(input())
length = 0
while n > 0:
    n //= 10  # это эквивалентно n = n // 10
    length += 1
print(length)
```

В этом цикле мы отбрасываем по одной цифре числа, начиная с конца, что эквивалентно целочисленному делению на 10 (n //= 10), при этом считаем в переменной length, сколько раз это было сделано.

В языке Питон есть и другой способ решения этой задачи: length = len(str(i)).

2.2 Инструкции управления циклом

После тела цикла можно написать слово else: и после него блок операций, который будет выполнен один раз после окончания цикла, когда проверяемое условие станет неверно:

```
i = 1
while i <= 10:
    print(i)
    i += 1
else:
    print('Цикл окончен, i =', i)
```

Казалось бы, никакого смысла в этом нет, ведь эту же инструкцию можно просто написать после окончания цикла. Смысл появляется только вместе с инструкцией break. Если во время выполнения Питон встречает инструкцию break внутри цикла, то он сразу же прекращает выполнение этого цикла и выходит из него. При этом ветка else исполняться не будет. Разумеется, инструкцию break осмысленно вызывать только внутри инструкции if, то есть она должна выполняться только при выполнении какого-то особенного условия.

Приведем пример программы, которая считывает числа до тех пор, пока не встретит отрицательное число. При появлении отрицательного числа программа завершается. В первом варианте последовательность чисел завершается числом 0 (при считывании которого надо остановиться).

```
a = int(input())
while a != 0:
    if a < 0:
        print('Встретилось отрицательное число', a)
        break
    a = int(input())
else:
    print('Ни одного отрицательного числа не встретилось')
```

Во втором варианте программы сначала на вход подается количество элементов последовательности, а затем и сами элементы. В таком случае удобно воспользоваться циклом for. Цикл for также может иметь ветку else и содержать инструкции break внутри себя.

```
n = int(input())
for i in range(n):
    a = int(input())
    if a < 0:
        print('Встретилось отрицательное число', a)
        break
else:
    print('Ни одного отрицательного числа не встретилось')
```

Другая инструкция управления циклом – continue (продолжение цикла). Если эта инструкция встречается где-то посередине цикла, то пропускаются все оставшиеся инструкции до конца цикла, и исполнение цикла продолжается со следующей итерации.

Если инструкции break и continue содержатся внутри нескольких вложенных циклов, то они влияют лишь на исполнение самого внутреннего цикла. Вот не самый интеллектуальный пример, который это демонстрирует:

```
for i in range(3):
    for j in range(5):
        if j > i:
            break
        print(i, j)
```

Увлечение инструкциями break и continue не поощряется, если можно обойтись без их использования. Вот типичный пример плохого использования инструкции break (данный код считает количество знаков в числе).

```
n = int(input())
length = 0
while True:
    length += 1
    n //= 10
    if n == 0:
        break
print('Длина числа равна', length)
```

Гораздо лучше переписать этот цикл так:

```
n = int(input())
length = 0
while n != 0:
    length += 1
    n //= 10
print('Длина числа равна', length)
```

Впрочем, на Питоне можно предложить и более изящное решение:

```
n = int(input())
print('Длина числа равна', len(str(n)))
```

2.3 Множественное присваивание

В Питоне можно за одну инструкцию присваивания изменять значение сразу нескольких переменных. Делается это так:

```
a, b = 0, 1
```

Этот код можно записать и так:

```
a = 0
b = 1
```

Отличие двух способов состоит в том, что множественное присваивание в первом способе меняет значение двух переменных одновременно.

Если слева от знака «=» в множественном присваивании должны стоять через запятую имена переменных, то справа могут стоять произвольные выражения, разделённые запятыми. Главное, чтобы слева и справа от знака присваивания было одинаковое число элементов.

Множественное присваивание удобно использовать, когда нужно обменять значения двух переменных. В обычных языках программирования без использования специальных функций это делается так:

```
a = 1
b = 2
tmp = a
a = b
b = tmp
```

```
print(a, b)
# 2 1
```

В Питоне то же действие записывается в одну строчку:

```
a = 1
b = 2
a, b = b, a
print(a, b)
# 2 1
```

3 Порядок выполнения работы

Получить задание для выполнения лабораторной работы (раздел 4) согласно своему варианту (по журналу). Разработать программу.

4 Задания для выполнения работы

Задание 1. Список квадратов.

По данному целому числу N распечатайте все квадраты натуральных чисел, не превосходящие N, в порядке возрастания.

Задание 2. Минимальный делитель.

Дано целое число, не меньшее 2. Выведите его наименьший натуральный делитель, отличный от 1.

Задание 3. Степень двойки.

По данному натуральному числу N найдите наибольшую целую степень двойки, не превосходящую N. Выведите показатель степени и саму степень.

Операцией возведения в степень пользоваться нельзя!

Задание 4. Утренняя пробежка.

В первый день спортсмен пробежал x километров, а затем он каждый день увеличивал пробег на 10% от предыдущего значения. По данному числу y определите номер дня, на который пробег спортсмена составит не менее y километров.

Программа получает на вход действительные числа x и y и должна вывести одно натуральное число.

Задание 5. Длина последовательности.

Программа получает на вход последовательность целых неотрицательных чисел, каждое число записано в отдельной строке. Последовательность завершается числом 0, при считывании которого программа должна закончить свою работу и вывести количество членов последовательности (не считая завершающего числа 0). Числа, следующие за числом 0, считывать не нужно.

Задание 6. Сумма последовательности.

Определите сумму всех элементов последовательности, завершающейся числом 0. В этой и во всех следующих задачах числа, следующие за первым нулем, учитывать не нужно.

Задание 7. Среднее значение последовательности.

Определите среднее значение всех элементов последовательности, завершающейся числом 0.

Задание 8. Максимум последовательности.

Последовательность состоит из натуральных чисел и завершается числом 0. Определите значение наибольшего элемента последовательности.

Задание 9. Индекс максимума последовательности.

Последовательность состоит из натуральных чисел и завершается числом 0. Определите индекс наибольшего элемента последовательности. Если наибольших элементов несколько, выведите индекс первого из них. Нумерация элементов начинается с нуля.

Задание 10. Количество четных элементов последовательности.

Определите количество четных элементов в последовательности, завершающейся числом 0.

Задание 11. Количество элементов, которые больше предыдущего.

Последовательность состоит из натуральных чисел и завершается числом 0. Определите, сколько элементов этой последовательности больше предыдущего элемента.

Задание 12. Второй максимум.

Последовательность состоит из различных натуральных чисел и завершается числом 0. Определите значение второго по величине элемента в этой последовательности. Гарантируется, что в последовательности есть хотя бы два элемента.

Задание 13. Количество элементов, равных максимуму.

Последовательность состоит из натуральных чисел и завершается числом 0. Определите, сколько элементов этой последовательности равны ее наибольшему элементу.

Задание 14. Числа Фибоначчи.

Последовательность Фибоначчи определяется так:

$$\varphi_0 = 0,\ \varphi_1 = 1,\ \varphi_n = \varphi_{n-1} + \varphi_{n-2}.$$

По данному числу n определите n-е число Фибоначчи φ_n.

Эту задачу можно решать и циклом for.

Задание 15. Номер числа Фибоначчи.

Дано натуральное число A. Определите, каким по счету числом Фибоначчи оно является, то есть выведите такое число n, что $\varphi_n = A$. Если A не является числом Фибоначчи, выведите число -1.

Задание 16. Максимальное число идущих подряд равных элементов.

Дана последовательность натуральных чисел, завершающаяся числом 0. Определите, какое наибольшее число подряд идущих элементов этой последовательности равны друг другу.

Задание 17. Стандартное отклонение.

Дана последовательность натуральных чисел $x_1, x_2, \ldots, x_n$. Стандартным отклонением называется величина

$$\sigma = \sqrt{\frac{(x_1 - s)^2 + (x_2 - s)^2 + \ldots + (x_n - s)^2}{n - 1}}$$

где $s = \frac{x_1 + x_2 + \cdots + x_n}{n}$ – среднее арифметическое последовательности.

Определите стандартное отклонение для данной последовательности натуральных чисел, завершающейся числом 0.