Практическая работа 4

Циклы

Цикл while

Цикл while используется в Python для неоднократного исполнения определенной инструкции до тех пор, пока заданное условие остается истинным. Этот цикл позволяет программе перебирать блок кода.

```
while test_expression:
    body of while
```

Сначала программа оценивает условие цикла while. Если оно истинное, начинается цикл, и тело while исполняется. Тело будет исполняться до тех пор, пока условие остается истинным. Если оно становится ложным, программа выходит из цикла и прекращает исполнение тела.

Рассмотрим пример, чтобы лучше понять.

```
a = 1
while a < 10:
   print('Цикл выполнился', a, 'раз(а)')
   a = a+1
print('Цикл окончен')
```

Бесконечный цикл while в Python

Бесконечный цикл while — это цикл, в котором условие никогда не становится ложным. Это значит, что тело исполняется снова и снова, а цикл никогда не заканчивается.

Следующий пример — бесконечный цикл:

```
a = 1
while a==1:
   b = input('Как тебя зовут?')
   print('Привет', b, ', Добро пожаловать')
```

Else в цикле while

В Python с циклами while также можно использовать инструкцию else. В этом случае блок в else исполняется, когда условие цикла становится ложным.

```
a = 1


while a < 5:
  print('условие верно')
  a = a + 1
else:
  print('условие неверно')
```

Программа исполняет код цикла while до тех, пока условие истинно, то есть пока значение a меньше 5. Поскольку начальное значение a равно 1, а с каждым циклом оно увеличивается на 1, условие станет ложным, когда программа доберется до четвертой итерации — в этот момент значение a изменится с 4 до 5. Программа проверит условие еще раз, убедится, что оно ложно и исполнит блок else, отобразив «условие неверно».

Прерывания цикла while в Python

В Python есть два ключевых слова, с помощью которых можно преждевременно остановить итерацию цикла.

Break — ключевое слово break прерывает цикл и передает управление в конец цикла

```
a = 1
while a < 5:
   a += 1
   if a == 3:
        break
   print(a) # 2
```

Continue — ключевое слово continue прерывает текущую итерацию и передает управление в начало цикла, после чего условие снова проверяется. Если оно истинно, исполняется следующая итерация.

```
a = 1


while a < 5:
    a += 1
    if a == 3:
        continue
    print(a)  # 2, 4, 5
```

Python цикл for — for i in range

Циклы for в Python

Циклы for повторяют определённый код для некоторого набора значений.

Из документации Python можно узнать, что в нем циклы for работают несколько иначе, чем в таких языках, как JavaScript или C.

Цикл for присваивает итерируемой переменной каждое значение из предоставленного списка, массива или строки и повторяет код в теле цикла for для каждого установленного таким образом значения переменной-итератора.

В приведенном ниже примере мы используем цикл for для вывода каждого числа в нашем массиве.

КОПИРОВАТЬ

```
# Простой пример цикла for
for i in [0, 1, 2, 3, 4, 5]:
    print(i, end="; ") # выведет: 0; 1; 2; 3; 4; 5;
```

В тело цикла for можно включить и более сложную логику. В следующем примере мы выводим результат небольшого вычисления, основанного на значении переменной i.

```
# Пример посложнее
for i in [0, 1, 2, 3, 4, 5]:
    x = (i-2)*(i+2) - i**2 + 4
    print(x, end="; ") # выведет: 0; 0; 0; 0; 0; 0;
```

Когда значения в массиве для нашего цикла for представляют собой некоторую закономерную последовательность, мы можем использовать функцию Python range() вместо того, чтобы вписывать содержимое нашего массива вручную.

Функция Range в Python

Функция range() возвращает последовательность целых чисел на основе переданных ей аргументов. Дополнительную информацию можно найти в документации Python по функции range().

```
range(stop)
range(start, stop[, step])
```

Аргумент start — это первое значение в диапазоне. Если функция range() вызывается только с одним аргументом, то Python считает, что start = 0.

Аргумент stop — это верхняя граница диапазона. Важно понимать, что само граничное значение не включается в последовательность.

В примере ниже у нас есть диапазон, начинающийся со значения по умолчанию, равному 0, и включающий целые числа меньше 6.

```
# Использование range() с единственным аргументом
for i in range(6):
    print(i, end="; ") # выведет: 0; 1; 2; 3; 4; 5;
```

В следующем примере мы задаем start = -2 и включаем целые числа меньше 4.

```
# В этот раз передаются два аргумента
for i in range(-2, 4):
    print(i, end="; ") # выведет: -2; -1; 0; 1; 2; 3;
```

Необязательное значение step (шаг) управляет приращением между значениями последовательности. По умолчанию step = 1.

В нашем последнем примере мы используем диапазон целых чисел от -2 до 6 и задаем step = 2.

```
# Здесь используются все аргументы range()
for i in range(-2, 6, 2):
    print(i, end="; ") # выведет: -2; 0; 2; 4;
```

Задачи

1. Даны два целых числа `A` и `B` (при этом `A ≤ B`). Выведите все числа от `A` до `B` включительно.
2. Даны два целых числа `A` и `B`. Выведите все числа от `A` до `B` включительно, в порядке возрастания, если `A < B`, или в порядке убывания в противном случае.
3. Даны два целых числа `A` и `B`, `A>B`. Выведите все нечётные числа от `A` до `B` включительно, в порядке убывания.
4. Дано несколько чисел. Вычислите их сумму. Сначала вводите количество чисел `N`, затем вводится ровно `N` целых чисел. Постройте решение так, чтобы использовалось минимальное количество переменных.
5. По данному натуральному n вычислите сумму `1^3+2^3+3^3+...+n^3`.
6. Факториалом числа `n` называется произведение `1 × 2 × ... × n`. Обозначение: `n!`. По данному натуральному `n` вычислите значение `n!`. Пользоваться математической библиотекой `math` в этой задаче запрещено.
7. По данному натуральном `n` вычислите сумму `1!+2!+3!+...+n!`. В решении этой задачи можно использовать только один цикл. Пользоваться математической библиотекой `math` в этой задаче запрещено.
8. По данному натуральному `n ≤ 9` выведите лесенку из `n` ступенек, `i`-я ступенька состоит из чисел от `1 до i` без пробелов.
9. Пользователь вводит число `N` с клавиатуры - количество чисел из ряда Фибоначчи. Посчитайте сумму этих чисел.

10. Пользователь вводит два числа `N` и `K` с клавиатуры. `N` - количество чисел из ряда Фибоначчи, `K` - порядковый номер в ряду, с которого нужно начать. Посчитайте сумму этих чисел. В решении этой задачи можно использовать только один цикл.