Дата: 2020/12/04
Предмет: Информатика
Тема: Элементы программирования
Тип занятия: Лекция
Группа: ТК-20, ТКБ-20

1. Теоретический материал

1.1. Инструкции циклов

Циклы применяются в том случае, если несколько раз подряд нужно выполнить одни и те же действия.

1.1.1. Цикл for

Цикл с параметром `for` применяется в том случае, когда количество повторений (итераций) нам известно заранее.

Определение 1. *Синтаксис цикла* `for`

```
for переменная in последовательность:
    команды тела цикла
else:
    команды, выполняемые после окончания цикла
```

Цикл перебирает элементы *последовательности*, присваивая их значения *переменной*. Каждое такое присваивание соответствует одной итерации цикла. На каждой итерации выполняются команды, записанные с отступом после двоеточия. Необязательная часть `else` может быть выполнена после окончания перебора элементов *последовательности*.

В качестве *последовательности* можно взять список, кортеж, словарь, объект `range` и множество (последнее – только в случае, когда не важно, в каком порядке будут перебраны элементы).

Отдельно рассмотрим использование `range`. Пример ниже показывает особенности этой функции.

Листинг 1. *Функция* `range`

```
n = 5
print(range(n))
print(type(range(n)))
print(list(range(n)))
```

Результат:

```
range(0, 5)
<class 'range'>
[0, 1, 2, 3, 4]
```

Видно, что команда `print` не выводит содержимое объекта `range`. После применения преобразования к типу `list` мы видим последовательность целых чисел, начинающаяся с 0 и заканчивающаяся числом $n-1$. Следующий пример показывает шире возможности `range`.

Листинг 2. *Функция* `range`

```
n = 10
print(list(range(n)))
print(list(range(3,n)))
print(list(range(2,n,2)))
print(list(range(n,1,-2)))
print(list(range(n-20,n+20,4)))
```

Результат:

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
[3, 4, 5, 6, 7, 8, 9]
[2, 4, 6, 8]
[10, 8, 6, 4, 2]
[-10, -6, -2, 2, 6, 10, 14, 18, 22, 26]
```

Приведём пример использования цикла с параметром и функцией `range`. Найдём конечную сумму $\sum\limits_{n=1}^{100} \frac{1}{n^2}$. Заметим, что для улучшения точности необходимо начать с самых маленьких слагаемых (то есть, с $n = 100$).

Листинг 3. *Вычисление конечной суммы*

```
mySum = 0
for n in range(100,0,-1):
    mySum = mySum + 1 / (n * n)
print(mySum)
```

Результат:

```
1.634983900184893
```

1.1.2. Цикл while

Цикл с условием `while` применяется в том случае, когда количество повторений (итераций) нам заранее не известно.

Определение 2. *Синтаксис цикла* `while`

```
while условие:
    команды тела цикла
else:
    команды, выполняемые после окончания цикла
```

Перед выполнением очередной итерации проверяется *условие*, и если оно истинно, то итерация выполняется (выполняются *команды тела цикла*), а если *условие* ложно, то цикл завершается. Если была указана необязательная часть `else`, то выполняются команды этой необязательной части.

Приведём пример использования цикла с условием для вычисления конечной суммы $\sum_{n=1}^{100} \frac{1}{n^2}$.

Листинг 4. *Вычисление конечной суммы*

```
mySum = 0
n = 100
while n>0:
    mySum = mySum + 1 / (n * n)
    n = n - 1
print(mySum)
```

Результат:

```
1.634983900184893
```

Заметим, что изменение переменной в результате арифметической операции можно записывать более кратко:

Длинная запись	Короткая запись
`n = n + c`	`n += c`
`n = n - c`	`n -= c`
`n = n * c`	`n *= c`
`n = n / c`	`n /= c`
`n = n // c`	`n //= c`
`n = n % c`	`n %= c`
`n = n ** c`	`n **= c`

1.1.3. Операторы break и continue

Оператор `break` используется для того, чтобы досрочно прервать выполнение цикла. В последующем пункте будет показан пример использования данного оператора.

Оператор `continue` начинает новую итерацию цикла, минуя команды, записанные после этого оператора в теле цикла.