Примеры работы оператора цикла for. Функция enumerate()

Мы продолжаем знакомиться с оператором цикла for языка Python. Сегодня мы с вами рассмотрим несколько примеров использования цикла for при решении разных задач. И первая задача – вычислить факториал от натурального числа n. Пусть пользователь вводит натуральное целое число, а мы будем выдавать значение его факториала:

```
n = int(input("Введите натурально

if n < 1 or n > 100:
    print("Неверно введено натура
else:
    p = 1
    for i in range(1, n+1):
        p *= i

    print(f"Факториал {n}! = {p}"
```

Обратите внимание на запись функции range(). Чтобы она выдавала последовательность целых чисел от 1 до n включительно, необходимо записать n+1, так как последнее значение не включается в диапазон.

Следующий пример – отображение символов * в виде елочки. Сделать это можно очень просто:

```
for i in range(1, 7):
    print('*' * i)
```

```
*
**
***
****
*****
******
```

Здесь счетчик i будет принимать значения 1, 2, 3, 4, 5 и 6. Соответственно, функция print() будет дублировать символ * i раз. В итоге, в первой строке будет одна звездочка, во второй – две и так до шести звездочек. Видите, как просто можно реализовать такую программу, используя оператор цикла и оператор работы со строками. Именно поэтому, все, что мы проходим, нужно хорошо запоминать и применять по мере необходимости.

Давайте, теперь, соединим все слова списка в одно предложение. Предположим, что у нас имеется список со словами: `words = ["Python", "дай", "мне",`

Которые объединим через цикл for:

```
s = ''
for w in words:
    s += ' ' + w

print(s)
```

У нас здесь получается вначале строки пробел. Убрать его можно двумя способами. Либо вызвать метод строки: `s = s.lstrip()`

А вот внимательный ученик с упреком заявил бы, что все это можно сделать еще проще – с помощью уже знакомого нам метода join():

```
print(" ".join(words))
```

И будет абсолютно прав! Те из вас, кто думал также – просто красавчики! Так держать!

Следующая задача. В списке: `digs = [4, 3, 100, -53, -30, 1, 3`

все двузначные числа заменить нулями. Используя те знания, что у нас сейчас есть, это можно сделать так:

```
for i in range(len(digs)):
    if 10 <= abs(digs[i]) <= 99:
        digs[i] = 0

print(digs)
```

Обратите внимание, что мы здесь в цикле перебираем индексы элементов списка, а не сами элементы. Благодаря этому, в цикле получаем возможность изменять значение i-го элемента.

Как видите, в этой задаче нам потребовалось в цикле знать и индекс элемента и его значение. Именно для таких случаев в Python существует специальная функция: **индекс, значение = enumerate(объект)**

которая возвращает пару (индекс, значение). Используя эту функцию, можно переписать наш пример, следующим образом:

```
for i, d in enumerate(digs):
    if 10 <= abs(d) <= 99:
        digs[i] = 0
```

Для меня такой вариант выглядит более читабельным и удобным. Во всем остальном эта программа подобна предыдущему варианту.

Последний пример, который я приведу на этом занятии –преобразование кириллицы в латиницу. Такая задача иногда возникает при создании сайтов. Вначале определим список замен для соответствующих русских букв (по алфавиту от а до я):

```
t = ['a', 'b', 'v', 'g', 'd', 'e'
     'z', 'i', 'y', 'k', 'l', 'm'
     'r', 's', 't', 'u', 'f', 'h'
     'shch', '', 'y', '', 'e', 'y
]
```

Затем, определим кодовое значение для первой буквы этого списка:

```
start = ord('a')
```

Далее, зададим строку и переменную slug, где будем формировать строку на латинице:

```
title = "Программирование на Python"
slug = ''
```

Затем, само преобразование в цикле, перебирая каждый символ исходной строки (причем, предварительно, строку мы переводим в нижний регистр – все буквы становятся малыми):

```
for s in title.lower():
    if 'а' <= s <= 'я':
        slug += t[ord(s) - start]
   elif s == 'ё':
        slug += 'yo'
    elif s in ' !?:,.':
        slug += '-'
    else:
        slug += s
```

После преобразования следует удалить все подряд идущие символы дефиса:

```
while slug.count('--'):
   slug = slug.replace('--', '-')

print(slug)
```