СОКРАЩЕНИЯ В PYTHON или однострочники

Циклы с вложенными условиями, иерархия условных конструкций, как правило приводят к значительному количеству строчек в коде.

Читаемость последнего снижется. На помощь приходят приемы сокращений. Или - однострочники.

Ниже приведем сравнительный анализ кода в полном формате и в записи однострочником.

Пример 1. Фрагмент классического кода:

```
if a < 3:
    b = 5
elif a == 4:
    b = 9
else:
    b = 0
```

Замена кода выше однострочником:

```
b = 5 if a < 3 else (b = 9 if a == 4 else 0)
```

СОКРАЩЕНИЯ В PYTHON или однострочники

В случае с циклами такой прием также работает. Например, есть список (list).

Допустим задача перенести значения данного списка в другой, умножив каждый из элементов на два.

Пример 2. Фрагмент классического кода:

```
fst = [2, 4, 6, 8]
scnd = []
for item in fst:
    scnd.append (item * 2)
```

Замена кода выше однострочником:

```
scnd = [item * 2 for item in fst]
```

Как мы можем заметить, всегда есть возможность оптимизации кода! И, если словосочетанию ***clear coding*** уделять серьезное значение, всегда найдется то, что можно упростить и заменить однострочником.

СОКРАЩЕНИЯ В PYTHON или однострочники

Усложним задачу, добавив условие в цикл. Например, во второй список попадают только нечетные значения из первого. В этом случае необходимо вводить дополнительную конструкцию с **if**.

Пример 3. Полный листинг кода:

```
num = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
res = []

for item in num:
    if item % 2:
        res.append(item)
print (res)
```

Замена фрагмента кода выше, используя однострочник (вновь обошлись без **append**):

```
res = [item for item in num if item % 2]
    print (res)
```

Строк меньше, результат – тот же.

СОКРАЩЕНИЯ В PYTHON или однострочники

И последнее. При изучении языка программирования (не только Python), следует обязательно изучать все возможности приемов и методов.

Рассмотрим последний пример (не совсем однострочник, но – упрощение):

Пример 4. Вывод имен через запятую:

```
names = ["Петя", "Митя", "Сеня", "Витя"]
print(', '.join(str(x) for x in names))
```

Оказывается то же самое можно сделать, не используя ни **join**, ни цикл **for**:

```
names = ["Петя", "Митя", "Сеня", "Витя"]
print(*names, sep=', ')
```

Конечно, как и во всем, никогда не следует переусердствовать, пытаясь использовать однострочники повсюду. Это также сделает код неопрятным и не читабельным.

Мера хороша также и в кодировании.