Методические указания

Урок 18.1. Генераторы и итераторы

Задачи урока:

- Генераторы и итераторы

0. Подготовка к уроку

До начала урока преподавателю необходимо:

1) Просмотреть, как ученики справились с домашним заданием
2) Прочитать методичку

1. Списковые включения

Учитель: На сегодняшнем занятии мы начнем знакомиться с такой темой как генераторы и итераторы. Соответственно научимся работать с каждым из них и разберем для чего они нужны.
Выражение-генератор — это объект, выполняющий то же вычисление, что и списковое включение, но выдающий результат в итеративной форме.
Мы с вами еще не сталкивались, с таким понятием как списковое включение. Для начала давайте разберемся немного с ним, а потом перейдем к выражениям генераторам, ну а после и разберем функции генераторы.
Существуют списковые включения, включения множеств и словарей, которые позволяют нам создавать из определенного набора данных списки, множества и словари, соответственно.
Представим, что нам необходимо создать список, состоящий из всех чисел от 1 до 10. Как мы это можем сделать?

1. Заполнить вручную, но если элементов 100, 1000, 10000
2. Использовать цикл for
3. Списковое включение

```
# через цикл
list1 = []
for i in range(1, 11):
    list1.append(i)

print(list1)
```

```
# используя списковое включение
list1 = [i for i in range(1, 11)]

print(list1)
```

Согласитесь вариант через списковое включение более компактен. Давайте разберем синтаксис данного оператора.

```
[выражение for элемент in последовательность]
```

выражение - какое либо выражение преобразующее значение из цикла. Например мы хотим список квадратов от 1 до 10. Для создания спискового включения, используем соответственно квадратные скобки.

```
list1 = [i ** 2 for i in range(1, 11)]

print(list1)
```

или чтобы значения были строчные

```
list1 = [str(i) for i in range(1, 11)]

print(list1)
```

Еще пример. Нам необходимо создать список из целых чисел, но вводить числа мы должны сами. Через цикл, как реализовать, думаю понятно, а вот с помощью включения давайте рассмотрим

```
list1 = [int(i) for i in input().split()]

print(list1)
```

Разберем данное решение. В генераторе мы указываем цикл который будет работать пока мы вводим данные и не нажимаем на Enter. Для разбиения нашей строки на список по пробелам, мы используем split(), без указания своего разделителя(по умолчанию пробел). Так как с инпута мы получаем только строчный тип, в выражении мы преобразуем его в целое число с помощью функции int().

Немного усложним пример. предположим нам необходимо создать список только из четных чисел от 1 до 100. Как вы думаете можно решить данную задачу с использованием генераторов.
Правильный ответ: добавив в генератор условие. Давайте реализуем данное решение и убедимся, что оно работает.

```
[выражение for элемент in последовательность if условие]
```

```
list1 = [i for i in range(1, 101) if i % 2 == 0]

print(list1)
```

Условие в данном случае мы добавляем после цикла. Вначале выполняется цикл, в котором создается переменная i, далее i проходит через условие и после выполняется выражение. При запуске данного кода у нас создается список из четных элементов от 1 до 100 включительно. Согласитесь, что данный подход намного компактнее и удобнее.

При необходимости мы можем использовать и два цикла

```
list1 = [i * j for i in range(1, 10) for j in [1, 2, 3]]

print(list1)
```

Посмотрите на вывод данного кода и подумайте как работает данное включение.
А давайте добавим туда еще и условие, а лучше сразу два.

```
list1 = [i * j for i in range(1, 10) if i % 2 == 0 for j in [1, 2, 3] if j
!= 2]

print(list1)
```

Данный код конечно работает, но читаемость его уже сильно пострадала. Важно: не забывайте, краткость это хорошо, но читаемость должна быть на первом месте.

Давайте подумаем, а если нам необходимо создать с помощью включения не список, а кортеж. Как поступить?

Если вы подумали, что использовать круглые скобки, но это не так и о этом мы узнаем на следующих занятиях. Правильным ответом будем использование функции tuple()

```
tuple1= tuple([i for i in range(1, 10) if i % 2 != 0])

print(tuple1)
print(type(tuple1))
```

2. Словарные включения

Словарные включения во многом аналогичны списковым включениям. В простых случаях они состоят из выражения, цикла и итерируемого объекта, которые заключаются в фигурные скобки.

```
dict1 = {x: x**2 + 1 for x in range(5)}
print(dict1)
```

Давайте разберем выражение x: x**2, которое состоит из двух выражений: первое x - это выражение которое создает ключи элементов, второе x**2 - создает значения элементов. Казалось бы, что благодаря этому можно использовать две разные переменные и создавать очень сложные словари, но из-за того что добавление элементов с одинаковыми ключами приводит к перезаписи значений, подобные трюки не получаются:

```
dict1 = {x: y for x in 'ABC' for y in 'XYZ'}
print(dict1)
```

Данное включение выдает такой результат, потому что каждому ключу из 'ABC' может соответствовать только одно значение. Поэтому его аналогичная запись будет выглядеть так:

```
dict1 = {x: 'Z' for x in 'ABC'}
print(dict1)
```

А также можно для получения данного результата воспользоваться методом .fromkeys():

```
dict1 = {}.fromkeys('ABC', 'Z')
print(dict1)
```

но, если все-таки требуется использовать две переменные, то:

```
dict1 = {x: y for x, y in [('A', 0), ('B', 1), ('C', 2)]}

print(dict1)
```

Словарные включения могут содержать внутри другие генераторы. Частенько можно встретить примеры, где создаются словари у которых значениями являются списки:

```
dict1 = {x: [y for y in range(x, x + 3)] for x in range(4)}

print(dict1)
```

```
dict1 = {x: [y % 2 for y in range(10)] for x in 'ABC'}
print(dict1)
```

```
dict1 = {'ABCDE'[i]: [i % 2]*5 for i in range(5)}
print
```

```
dict1 = {x: {y: 0 for y in 'XYZ'} for x in 'ABC'}
print(dict1)
```

```
dict1 = {x: {y: x for y in 'XYZ'} for x in 'ABC'}
print(dict1)
```

Создаваемые словарные включения могут иметь и более сложный синтаксис:

```
dict1 = {x: {y: [z for z in range(z, z+ 2)] for y in 'XYZ'} for x, z in
zip('ABC', range(3))}
print(dict1)
```

Вложенные включения множеств, могут быть использованы для хранения уникальных элементов. Например, вот так можно посмотреть на все остатки от деления квадратов чисел на определенный список делителей:

```
dict1 = {i: {j**2 % i for j in range(1, 100)} for i in [4, 5, 8, 9]}
print(dict1)
```

Условия в словарных включениях применяются точно также

```
dict1 = {x: {y: 3 for y in 'ABCD' if y != x} for x in 'ABCD'}
print(dict1)
```

Условное выражение if... else... указывается перед объявлением цикла for:

```
dict1 = {x: 1 if x in 'ACE' else 0 for x in 'ABCDEF'}

print(dict1)
```

3. Включения множеств

Самое простое включение множеств состоит из выражения, объявления цикла и итерируемого объекта, которые заключены в фигурные скобки:

```
set1 =  {i**2 % 4 for i in range(10)}
print(set1)
```

Условие if указывается после объявления цикла и итерируемого объекта:

```
set1 =   {i for i in ['ab_1', 'ac_2', 'bc_1', 'bc_2'] if 'a' not in i}
print(set1)
```

Условное выражение if... else... указывается до объявления цикла:

```
set1 =   {'A' + i[1:] if i[0] == 'a' else 'B' + i[1:] for i in ['ab_1',
'ac_2', 'bc_1', 'bc_2']}
print(set1)
```

Конструкции if и if... else... могут быть использованы одновременно:

```
set1 =  {'A' + i[1:] if i[0] == 'a' else 'B' + i[1:] for i in ['ab_1',
'ac_2', 'bc_1', 'bc_2'] if i[1] == 'c'}
print(set1)
```

4. Решение задач

Задача 1

Написать списковое включение, которое добавляет в список все четные числа, разделенные нацело на 2, в диапазоне от 1 до 100

```
print([i // 2 for i in range(1, 100) if i % 2 == 0])
```

Задача 2

Напишите функцию, которая принимает словарь с параметрами и возвращает строку запроса, сформированную из отсортированных в лексикографическом порядке параметров.

Пример:
Код print(query({'course': 'python', 'lesson': 2, 'challenge': 17})) должен возвращать строку:
challenge=17&course=python&lesson=2

Решение

```
def query(params):
      return '&'.join(sorted(f'{k}={v}' for k, v in params.items()))
```

Дополнительно

Если на уроке остается время, то ученикам можно предложить начать прорешивать домашнее задание.

Домашняя работа

Задача 1