Урок 3. Циклы в R

В этом уроке мы обсудим циклы. Продолжим пример с городами из прошлого урока. Создадим теперь два вектора городов: вектор `cities` из предыдущего урока, в который мы либо будем добавлять города, либо нет, и вектор `to.check` с перечнем городов, которые необходимо потестировать на принадлежность вектору `cities`.

```
cities <- c("Мурманск", "Киров", "Москва")
to.check <- c("Омск", "Москва", "Киров", "Калининград", "Пермь")
```

Для начала выведем на экран города из вектора `to.check` по очереди с новой строки. Для этого нам понадобится цикл `for`. Как вы уже знаете, циклы нужны для повторения одинаковых операций, чтобы избежать повторного запуска или копирования строк кода и следовать принципу программирования *don't repeat yourself* («не повторяйся»).

Как и в условных конструкциях, при построении циклов в R, в отличие от Python, вместо отступов используются фигурные скобки. В строке с оператором for в круглых скобках указываем, по чему итерируем («пробегаемся»), а затем в фигурных скобках прописываем тело цикла — набор действий, которые нужно повторить в цикле.

```
for (i in to.check){
print(i)
}
```

```
## [1] "Омск"
## [1] "Москва"
## [1] "Киров"
## [1] "Калининград"
## [1] "Пермь"
```

Здесь `i` означает любой элемент вектора `to.check` (итерируем по этому вектору).

В Python такой код выглядел бы следующим образом:

```
for i in to.check:
    print(i)
```

Теперь попробуем объединить цикл `for` и условную конструкцию из предыдущего урока, чтобы решить более сложную задачу: пройтись по всему вектору `to.check` и если город из этого вектора отсутствует в `cities`, добавить его туда.

```
for (i in to.check){
  if (i %in% cities){
print("No need to add")
}else{
cities <- c(cities, i)}
}
```

```
## [1] "No need to add"
## [1] "No need to add"
```

```
cities  # получилось
```

```
## [1] "Мурманск"    "Киров"       "Москва"      "Омск"        "Калининград"
## [6] "Пермь"
```

На Python аналогичный код выглядел бы так:

```
for i in to.check:
```

```
    if i in cities:
        print("No need to add")
    else:
        cities.append(i)
cities
```

Такая же схема, что и в предыдущем примере, однако теперь внутри цикла уже целая конструкция. Кроме того, на этом примере хороша видна разница между оператором `in` для итерирования и оператором `%in%` для принадлежности элемента вектору.

Для полноты материала про циклы нам осталось обсудить цикл `while`, однако стоит отметить, что в R он используется крайне редко. Цикл `while` достаточно медленный; к тому же, из-за того, что в R многие функции векторизованы, большой необходимости в нем на базовом уровне нет. Но, тем не менее, он может понадобится, если мы столкнемся с более продвинутой задачей, например, написать алгоритм для собственной модели, потому что существующие в R нас не устраивают.

Допустим, в переменной `err` сохранено значение ошибки предсказания модели, и мы хотим обучать модель до тех пор, пока ошибка не станет меньше 0.05. Строить модель, мы сейчас, конечно, не будем, вместо этого будем выводить строку "Model training" и уменьшать ошибку на 0.005.

```
err <- 0.1

while (err >= 0.05){
  print("Model training")
  err <- err - 0.005
  print(err)
}
```

```
## [1] "Model training"
## [1] 0.095
## [1] "Model training"
## [1] 0.09
## [1] "Model training"
## [1] 0.085
## [1] "Model training"
## [1] 0.08
## [1] "Model training"
## [1] 0.075
## [1] "Model training"
## [1] 0.07
## [1] "Model training"
## [1] 0.065
## [1] "Model training"
## [1] 0.06
## [1] "Model training"
## [1] 0.055
## [1] "Model training"
## [1] 0.05
```

```
print("Training successful. Error is less than 0.05")
```

```
## [1] "Training successful. Error is less than 0.05"
```

В отличие от for, нам не нужен перечень элементов, по которому надо двигаться; здесь мы задаем условие и говорим R, что набор действий нужно выполнять до тех пор, пока условие верно. Получается, пока ошибка слишком большая (не менее 0.05), мы выполняем тело цикла, операции, заключенные в фигурные скобки. На этом мы завершаем обсуждение циклов и самое время перейти к практическому заданию.

На этом мы завершаем обсуждение циклов и самое время перейти к практическому заданию.