Рекурсивні функції в PYTHON

Сиропятов Ігор, ФІТ 1-12

Рекурсія – тема в математиці та комп’ютерних науках. У мовах програмування, термін рекурсія відповідає функції, яка викликає себе. Інакше кажучи, це оголошення функції, що включає в себе тіло функції та її виклик. Можна випадково створити нескінченний цикл, який спричинить зависання програми. Це може статися, тому що в результаті використання рекурсії, функція може безкінечно викликати себе. Тому, як і з іншими потенційно нескінченними циклами, потрібно переконатись, що існує умова виходу з циклу. У більшості рекурсивних функцій, ідея полягає в тому, щоб розбити процедуру на менші фрагменти, які можна опрацьовувати тією ж функцією.

Ліміт рекурсії в Python

На початку було згадано, що можна створити нескінченний рекурсивний цикл. Це можливо в деяких мовах програмування, але в Python є ліміт рекурсії. Це можна перевірити, виконавши

```
>>> import sys
>>> sys.getrecursionlimit()
1000
```

Якщо Ви вважаєте, що ліміт є занадто малим для Вашої програми, його можна встановити за допомогою функції ***setrecursionlimit()*** модуля ***sys***. Спробуймо створити рекурсивну функцію, яка перевищить ліміт, щоб побачити, що відбудеться:

```
def recursive():
    recursive()

if __name__ == '__main__':
    recursive()
```

Якщо Ви запустите цей код, то повинні побачити такий виняток: ***RuntimeError: maximum recursion depth exceeded.*** Python запобігає створенню функції, яка вилиється у нескінченний рекурсивний цикл.

Декомпозиція списків за допомогою рекурсії

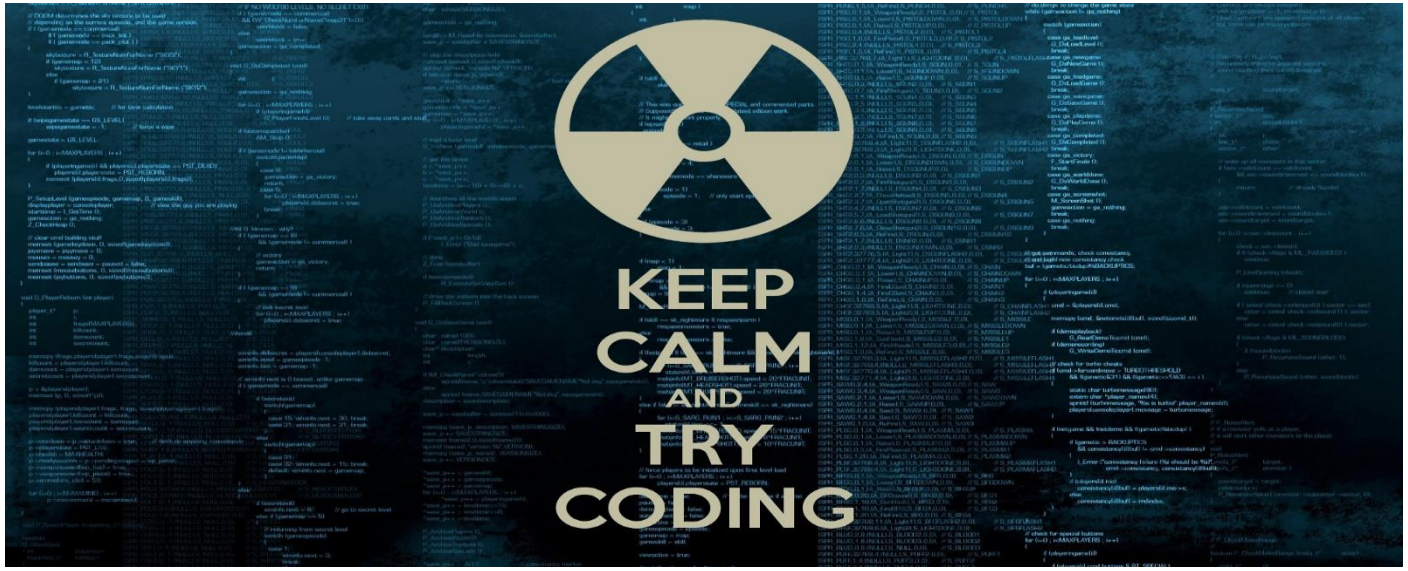
Тим не менше, є й інші речі, які можна робити за допомогою рекурсії, окрім обчислення факторіалів. Більш практичною ілюстрацією було б, наприклад, створення функції для розкладання вкладеного списку на складові:

```
# flatten.py

def flatten(a_list, flat_list=None):
    if flat_list is None:
        flat_list = []

    for item in a_list:
        if isinstance(item, list):
            flatten(item, flat_list)
        else:
            flat_list.append(item)

    return flat_list

if __name__ == '__main__':
    nested = [1, 2, 3, [4, 5], 6]
    x = flatten(nested)
    print(x)
```

У результаті виконання цього коду, повинен бути список цілих чисел, а не список цілих чисел і ще один список. Звичайно, є й багато інших ефективних шляхів розкласти вкладений список на складові, такий як використання ***itertools.chain()***. Можливо Ви захочете поглянути на код класу ***chain()***, так як у ньому застосовується зовсім інший підхід для декомпозиції списку.

Висновок

Тепер ви можете мати базове розуміння того, як працює рекурсія та як ви можете використовувати її в Python. Я думаю, чудово, що Python має вбудований ліміт рекурсії, щоб запобігти розробникам створювати погано побудовані рекурсивні функції. Я також хочу зазначити, що за багато років роботи розробником навряд коли-небудь дійсно потрібно буде використовувати рекурсію для вирішення проблеми. Я впевнений, що існує безліч проблем, при яких рішення може бути реалізовано в рекурсивній функції, але у Python так багато інших способів зробити те саме, що я відчував потреби робити це. Також можна зазначити, що рекурсію може бути важко налагодити, оскільки важко сказати, якого рівня рекурсії ви досягли, коли сталася помилка.



Дякую за Увагу!