Python Advanced

Елементи функціонального програмування

Після уроку обов'язково

Повторіть цей урок у форматі відео на ITVDN.com

Перевірте, як Ви засвоїли цей матеріал на TestProvider.com

Тема

Елементи функціонального програмування

Зміст уроку

1. Функції як об'єкти першого класу (first-class citizens).
2. Лямбда-вирази.
3. Замикання.
4. Функції вищого порядку, каррування функцій.
5. Декоратори.
6. Функції filter, map і reduce.
7. Модулі functools, operator та itertools.

Поняття функціонального програмування

Функціональне програмування – розділ дискретної математики та парадигма програмування, у якій процес обчислення сприймається як обчислення значень функцій у математичному розумінні останніх (на відміну від функцій як підпрограм у процедурному програмуванні).

Функціональне програмування передбачає використання обчислення результатів функцій від вихідних даних і результатів інших функцій. Не передбачає явного зберігання стану програми. Відповідно, не передбачає і зміни цього стану (на відміну від імперативного, де однією з базових концепцій є змінна, що зберігає своє значення та дає змогу змінювати його з виконанням алгоритму).

Характерні риси функціонального програмування

- Розв'язання задачі записується як сукупність незалежних від зовнішнього стану функцій.
- Функції як об'єкти першого класу.
- Іммутабельність (незмінність) даних.
- Використання функцій вищого порядку.
- Каррування та часткове застосування функцій.

Функція як об'єкт першого класу

Об'єкт називають «**об'єктом першого класу»,** якщо він:

- може бути збережений у змінній чи структурах даних;
- може бути переданий у функцію як аргумент;
- може бути повернений із функції як результат;
- може бути створений під час виконання програми;
- незалежний від назви.

Термін «об'єкт» використовується тут у загальному розумінні та не обмежується об'єктами мови програмування.

У Python, як і функціональних мовах, функції є об'єктами першого класу.

Лямбда-вирази

Звичайне оголошення функції :

```
def add(x, y):
    return x + y
```

Використання лямбда-виразів (лямбда-функції, анонімної функції):

```
add = lambda x, y: x + y
```

Замикання

- **Замикання** (англ. **closure**) у програмуванні – функція, у тілі якої є посилання на змінні, які оголошені поза тілом цієї функції у навколишньому коді та які не є її параметрами.
- У разі замикання посилання на змінні зовнішньої функції дійсні всередині вкладеної функції доти, доки працює вкладена функція, навіть якщо зовнішня функція закінчила роботу, і змінні вийшли з області видимості.
- Замикання пов'язує код функції з її лексичним оточенням (місцем, де вона визначена в коді). Лексичні змінні замикання відрізняються від глобальних змінних тим, що де вони займають глобальне простір імен. Від змінних в об'єктах вони відрізняються тим, що прив'язані до функцій, а не об'єктів.
- У Python будь-які функції (зокрема й лямбда-вирази), які оголошені всередині інших функцій, є повноцінними замиканнями**.**

Опції вищого порядку

Функція вищого порядку – функція, яка приймає як аргументи інші функції або повертає іншу функцію як результат. Основна ідея полягає в тому, що функції мають той самий статус, що й інші об’єкти даних.

Каррування, або **Каррінг** (*англ. currying*) – перетворення функції від багатьох аргументів на функцію, яка бере свої аргументи по одному. Це перетворення було запроваджено М. Шейнфінкелем і Г. Фреге й отримало свою назву на честь Г. Каррі.

Декоратори

Декоратор Python – функція, яка приймає як параметр іншу функцію (або клас) і повертає нову, модифіковану функцію (або клас), яка її замінює.

Окрім цього, поняття функцій вищого порядку часто застосовується і для створення декораторів: часто потрібно, щоб декоратор приймав ще якісь параметри, окрім об'єкта, що модифікується. У такому випадку створюється функція, що створює та повертає декоратор, а під час застосування декоратора замість вказівки імені функції-декоратора ця функція викликається.

map, filter, reduce

Трьома класичними функціями вищого порядку, що з'явилися ще в мові програмування Lisp, які приймають функцію та послідовність, є map, filter та reduce.

- **map** застосовує функцію до кожного елемента послідовності. У Python 2 повертає список, а в Python 3 – об'єкт-ітератор.
- **filter** залишає лише ті елементи послідовності, для яких задана функція істинна. У Python 2 повертає список, а в Python 3 – об'єкт-ітератор.
- **reduce** (у Python 2 вбудована, а в Python 3 розташована в модулі functools) приймає функцію від двох аргументів, послідовність та опціональне початкове значення й обчислює згортку (fold) послідовності як результат послідовного застосування цієї функції до поточного значення (так званого акумулятора) і наступного елемента послідовності.

Модуль functools

Модуль *functools* містить велику кількість стандартних функцій вищого порядку. Серед них особливо корисні:

- **reduce** – розглянута вище;
- **lru_cache** – декоратор, що кешує значення функцій, які не змінюють свій результат за незмінних аргументів; корисний для кешування даних, мемоізації (збереження результатів для повернення без обчислення функції) значень рекурсивних функцій (наприклад, такого типу, як-от функція обчислення n-го числа Фібоначчі) тощо;
- **partial** – часткове застосування функції (виклик функції з меншою кількістю аргументів, ніж вона очікує, та отримання функції, яка приймає параметри, що залишилися).

Модуль itertools

Модуль *itertools* містить функції для роботи з ітераторами та створення ітераторів. Деякі з них:

- **product** – декартів добуток ітераторів (для уникнення вкладених циклів for);
- **permutations** – генерація перестановок;
- **combinations** – генерація сполучень;
- **combinations_with_replacement** – генерація розміщень;
- **chain** – з'єднання кількох ітераторів в один;
- **takewhile** – отримання значень послідовності, поки значення функції-предикату для її елементів істинне;
- **dropwhile** – отримання значень послідовності починаючи з елемента, для якого значення функції-предиката перестане бути істинним.

Модуль operator

Модуль *operator* містить функції, які відповідають стандартним операторам.

Приклади:

Функція модуля operator	Аналогічна лямбда-функція
add	lambda x, y: x + y
gt	lambda x, y: x > y
neg	lambda x: -x
…	

Q&A

Інформаційний відеосервіс для розробників програмного забезпечення

ITVDN

IT VIDEO DEVELOPERS NETWORK

TestProvider PROMETRIC TEST CENTER CyberBionic systematics Microsoft Partner Silver Learning Windows Azure Cloud Partner EBA European Business Association