Thực hành Python

Làm việc với kiểu List và xây dựng Function

Trần Quang Quý
e-mail: tqquy@ictu.edu.vn
Contact: 0818981166(zalo)

Trường Đại học Công nghệ Thông tin và Truyền thông Thái Nguyên

Tháng 09 năm 2021

Các bài tập luyện tập với LIST

List (Danh sách) là dãy mà có khả năng lưu giữ các kiểu dữ liệu khác nhau và có thể thay đổi. Trong Python, danh sách được viết trong dấu ngoặc vuông [ ]
Để khởi tạo một danh sách rỗng, chúng ta sử dụng: list1=[ ].
Trong kiểu dữ liệu danh sách, chúng ta có thể xóa, sửa, cập nhật và xóa toàn bộ danh sách. Các kiểu dữ liệu trong danh sách là khác nhau, cũng có thể là giống nhau (kiểu như mảng trong C).

Khái niệm và tính chất của LIST

Trong kiểu dữ liệu danh sách, một số phương thức cơ bản để thao tác cần ghi nhớ:

1. Phương thức list.append(x)
2. Phương thức list.extend(x)
3. Phương thức len()
4. Phương thức list.index(x) hoặc list.index(x,i)
5. Thao tác nối danh sách
6. Toán tử in và not in
7. Các phương thức cơ bản như sum, min, max.

Bài tập luyện tập với LIST và Function

Bài 1: Nhập từ bàn phím một số n nguyên dương, hãy tạo ra một danh sách là list1 với n phần tử nhập từ bàn phím là số nguyên. Yêu cầu: Hãy tính tổng số các phần tử chẵn và số các phần tử lẻ của list1.

Bài 2: Tổng k số hạng liên tiếp có tổng lớn nhất.

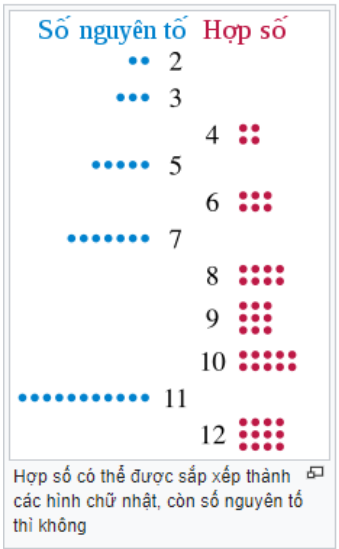
Nhập số nguyên dương $n, k(k \leq n)$ và danh sách gồm n số nguyên $a_0, a_1, ..., a_{n-1}$.

Tìm k phần tử liên tiếp trong dãy $a_i, a_{i+1}, ..., a_{i+k-1}$ đạt giá trị lớn nhất. Ví dụ dãy số sau: 1,2-19,3,4,-1 với $k = 3$ thì tổng 3 số hạng liên tiếp có tổng lớn nhấ là 3+4+(-1) = 6.

Gợi ý: Ở bài tập số 2 có thể sử dụng kỹ thuật dịch chuyển cửa sổ (subling windows)

Số nguyên tố - Prime number

Số nguyên tố là số tự nhiên lớn hơn 1 không phải là tích của hai số tự nhiên nhỏ hơn. Nói cách khác, số nguyên tố là những số chỉ có đúng hai ước số là 1 và chính nó. Các số tự nhiên lớn hơn 1 không phải là số nguyên tố được gọi là hợp số.



Hình 1: Số nguyên tố và hợp số

Bài 3: Sử dụng cách xây dựng hàm, hãy viết hàm kiểm tra số nguyên tố. Yêu cầu: Nhập vào một số nguyên dương n từ bàn phím, hãy liệt kê tất cả các số nguyên tố không vượt quá n.

Dãy số Fibonacci

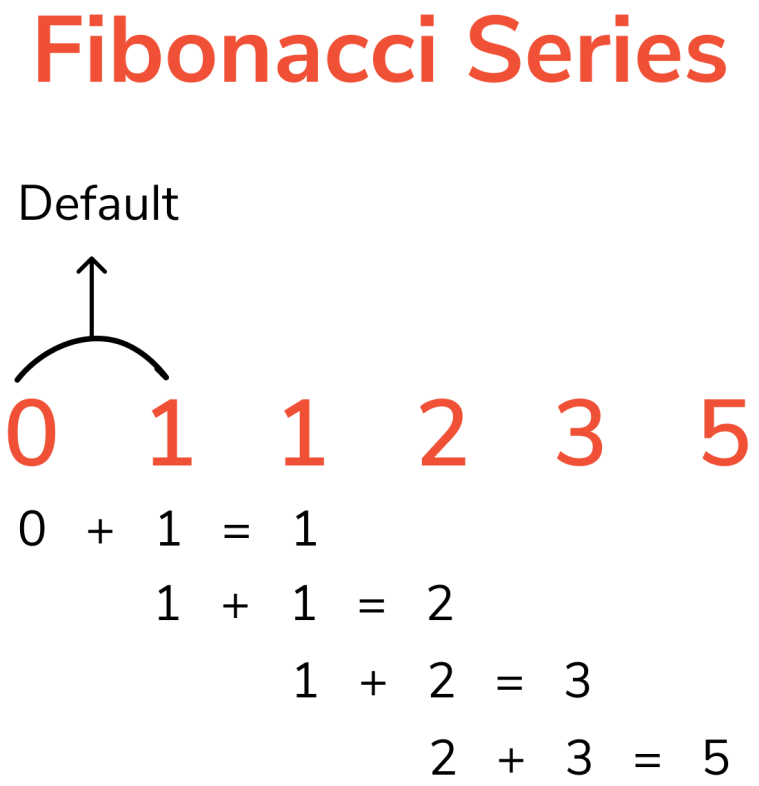
Dãy Fibonacci là dãy vô hạn các số tự nhiên bắt đầu bằng hai phần tử 0 và 1 hoặc 1 và 1, các phần tử sau đó được thiết lập theo quy tắc mỗi phần tử luôn bằng tổng hai phần tử trước nó. Công thức truy hồi của dãy Fibonacci là:

$$F(n) := \begin{cases} 1\,, & \text{khi } n = 1\,; \\ 1, & \text{khi } n = 2; \\ F(n-1) + F(n-2) & \text{khi } n > 2. \end{cases}$$

Hình 2: Truy hồi Fibonacci

Dãy số Fibonacci

Bài 4: Yêu cầu nhập vào một số nguyên dương n từ bàn phím, hãy in ra dãy số Fibonacci có chiều dài n Ví dụ: Nhập n = 6 thì in ra dãy là: 0,1,1,2,3,5.



Hình 3: Truy hồi Fibonacci

Bài 5: Chọn số

Cho dãy số gồm n số hạng $a_0, a_1, ..., a_{n-1}$. Hãy chọn các số hạng trong dãy sao cho thỏa mãn:

1. Không chọn hai số hạng kề nhau, tức là không chọn a_i và a_{i+1}.
2. Tổng các số hạng được chọn có giá trị lớn nhất.

Ví dụ: Dãy 1,-2,3,2,6. Ta chọn các số hạng 1,3,6.

Quy hoạch động - Dynamic Programming

Quy hoạch động là bài toán thường được sử dụng trong việc tối ưu và có sử dụng các bài toán con gối nhau, nếu một bài toán có các bài toán con gối nhau tức là các bài toán con được chia ra từ bài toán lớn được gọi lặp đi lặp lại thì khi đó chúng ta sử dụng quy hoạch động.
Tương tự như thuật toán chia để trị, quy hoạch động cũng chia bài toán lớn thành các bài toán con nhỏ hơn. Quy hoạch động được sử dụng khi các bài toán con này được gọi đi gọi lại. Phương pháp quy hoạch động sẽ lưu kết quả của bài toán con này, và khi được gọi, nó sẽ không cần phải tính lại, do đó làm giảm thời gian tính toán.

Dynamic Programming

Quy hoạch động sẽ không thể áp dụng được (hoặc nói đúng hơn là áp dụng cũng không có tác dụng gì) khi các bài toán con không gối nhau.
Một ví dụ rất điển hình của bài toán con gối nhau là bài toán tính số Fibonacci. Bài toán quá nổi tiếng rồi, chúng ta có thể tính toán số Fibonacci theo đúng công thức như sau:

```
def fib(n):
    if n <= 1:
        return n
    return fib(n -1) + fib(n - 2)
```

Hình 4: Ví dụ về quy hoạch động

Nếu tính toán như trên, chúng ta rất nhiều bài toán con sẽ được tính đi tính lại, điển hình là các số fib(0) và fib(1).

Và quy hoạch động chính là một trong số những phương pháp có thể giúp chúng ta tối ưu hóa quá trình tính toán này. Mỗi bài toán con (số fib) sẽ được lưu lại trước khi tính những bài toán con lớn hơn. Nhờ đó, mà việc tính toán giảm đi đáng kể, mỗi bài toán con chỉ cần tính đúng một lần.

Một số bài tập bổ trợ

Bài 6: Hãy nhập vào một số nguyên dương n từ bàn phím. In và liệt kê ra danh sách các ước số của n và đưa vào một danh sách có tên là list1.

Bài 7: Hãy viết hàm trả về giá trị in đảo ngược của một danh sách

Bài 8: Nhập lần lượt n số tự nhiên tạo thành một dãy $a_0, a_1, ..., a_n$. Gọi là dãy A

Hãy đưa ra một dãy con gọi là Bbao gồm các số nguyên tố của dãy A. In ra màn hình dãy B (Sử dụng sàng số nguyên tố Erastosthenes)

Bài 9: Cho trước một dãy A. Hãy in ra dãy B bằng cách tách tất cả các phần tử là số của A. Ví dụ: A =[1.2, "one", 0.15, "aA", "B", 1.5], thì dãy B sẽ là: B = [1.2,0.15,1.5]

Sàng số nguyên tố Erastosthenes

Sàng Eratosthenes là một thuật giải toán cổ xưa để tìm các số nguyên tố nhỏ hơn 100. Thuật toán này do nhà toán học cổ Hy Lạp là Eratosthenes (Ơ-ra-tô-xten) "phát minh" ra.

Ban đầu, nhà toán học Eratosthenes sau khi tìm ra thuật toán, đã lấy lá cọ và ghi tất cả các số từ 1 cho đến 100. Ông đã chọc thủng các hợp số và giữ nguyên các số nguyên tố. Bảng số nguyên tố còn lại trông rất giống một cái sàng. Do đó, nó có tên là sàng Eratosthenes.

Hình 5: Sàng số nguyên tố Erastosthenes

Thuật toán sàng số nguyên tố

Để tìm các số nguyên tố nhỏ hơn hoặc bằng số tự nhiên N bằng sàng Eratosthenes, ta làm như sau:

1. Tạo 1 danh sách các số tự nhiên liên tiếp từ 2 đến n: (2, 3, 4,. . . , n).
2. Giả sử tất cả các số trong danh sách đều là số nguyên tố. Trong đó, p = 2 là số nguyên tố đầu tiên.
3. Tất cả các bội số của p: 2p, 3p, 4p,. . . sẽ bị đánh dấu vì không phải là số nguyên tố.
4. Tìm các số còn lại trong danh sách mà chưa bị đánh dấu và phải lớn hơn p. Nếu không còn số nào, dừng tìm kiếm. Ngược lại, gán cho p giá trị bằng số nguyên tố tiếp theo và quay lại bước 3.
5. Khi giải thuật kết thúc, tất các số chưa bị đánh dấu trong danh sách là các số nguyên tố cần tìm.