Basic Python - For-Loop

Hoàng-Nguyên Vũ

1 Cấu trúc vòng lặp `for-loop` trong Python

Vòng lặp `for-loop` trong Python được sử dụng để lặp qua các phần tử trong một chuỗi (sequence) như danh sách, chuỗi ký tự, tuple, từ điển, hoặc dải giá trị số (`range`).

Cấu trúc chung

Cú pháp cơ bản của `for-loop` trong Python:

```
for variable in sequence:
    # Thực hiện các lệnh bên trong vòng lặp
```

Ví dụ 1: Lặp qua một danh sách

```
numbers = [1, 2, 3, 4, 5]
for num in numbers:
    print(num)
```

Ví dụ 2: Lặp qua một dải số

```
for i in range(5):  # range(5) tạo ra các số từ 0 đến 4
    print(i)
```

Ví dụ 3: Lặp qua một chuỗi ký tự

```
for char in "Python":
    print(char)
```

2 Bài tập

2.1 Tính lãi suất tiền gửi ngân hàng - Dùng vòng lặp for

+ **Số e** (còn gọi là hằng số Euler) là một số vô tỷ xuất hiện trong nhiều lĩnh vực toán học và khoa học, bao gồm cả lĩnh vực tài chính. Trong ngân hàng, số e được sử dụng để tính lãi suất kép, một phương pháp tính lãi suất trong đó tiền lãi được cộng vào số tiền gốc để tính lãi cho các kỳ hạn tiếp theo. Công thức tổng quát của số e như sau: $\lim_{n\to\infty}\left(1+\frac{1}{n}\right)^n$

+ Giả sử bạn có 1 Dollar, và bạn gởi vào ngân hàng và được **nhận lãi mỗi ngày**, vậy điều sẽ xảy ra nếu bạn gởi vào ngân hàng trong 1 năm thì tiền bạn nhận được là bao nhiêu ? Để trả lời câu hỏi này, chúng ta làm như sau:

  – Bước 1: Tiền nhận được sau 1 ngày gửi: $1+\frac{1}{365}$

  – Bước 2: lặp lại cách tính này cho 1 năm, ta sẽ được công thức của e: $\left(1+\frac{1}{n}\right)^n$

```
def compute_interest(money, period):
  ##### Your code here ######
```

Ví dụ:
- Test case 1: compute_interest(1, 12) → 2.613
- Test case 2: compute_interest(1, 365) → 2.714
- Test case 3: compute_interest(1, 730) → 2.716

2.2 Thuật toán sàng số nguyên tố Eratosthenes

Mô tả: Sử dụng thuật toán sàng Eratosthenes để kiểm tra xem một số nguyên n (được nhập từ người dùng) có phải là số nguyên tố hay không.

Algorithm 1 Kiểm tra số nguyên tố sử dụng sàng Eratosthenes

Require: Một số nguyên $n \geq 0$
Ensure: Trả về `True` nếu n là số nguyên tố, ngược lại trả về `False`

if $n < 2$ **then**
    return `False` ▷ Số nhỏ hơn 2 không phải là số nguyên tố
end if
Tạo một danh sách `prime` với $n + 1$ phần tử, tất cả được gán `True`
Gán `prime[0]` và `prime[1]` là `False` ▷ 0 và 1 không phải số nguyên tố
for $i \leftarrow 2$ **to** $\sqrt{n}$ **do**
    if `prime[i]` = `True` **then**
        ▷ Đánh dấu tất cả các bội số của i từ i^2 đến n
        for $j \leftarrow i^2$ **to** n **step** i **do**
            `prime[j]` $\leftarrow$ `False` ▷ j là bội số của i, không phải số nguyên tố
        end for
    end if
end for
return `prime[n]` ▷ Kiểm tra giá trị của `prime[n]`

```python
def is_prime_eratosthenes(n):
    ##### Your code here ######
```

Ví dụ:
- Test case 1: is_prime_eratosthenes(7) → True
- Test case 2: is_prime_eratosthenes(10) → False
- Test case 3: is_prime_eratosthenes(2) → True
- Test case 4: is_prime_eratosthenes(1) → False