

Loop

Kittikun Jitpaired

Introduction to Loops

Loops in Python provide a mechanism for executing a set of instructions repeatedly. They are fundamental constructs in programming that allow for efficient handling of iterative tasks. Python offers two primary types of loops: 'for' loops and 'while' loops, each serving different purposes in code execution.

The for Loop

The for loop in Python is designed to iterate over a sequence (such as a list, tuple, string, or range) or other iterable objects. It executes a set of statements once for each item in the sequence.

Basic syntax:

```
for item in sequence:  
    # code to be executed
```

Example:

```
fruits = ["apple", "banana", "cherry"]  
for fruit in fruits:  
    print(fruit)
```

```
apple  
banana  
cherry
```

The 'for' loop can also be used with the range() function to execute a set of statements a specific number of times:

```
for i in range(5):  
    print(i)
```

0
1
2
3
4

The while Loop

The while loop executes a set of statements as long as a given condition is true. It's particularly useful when the number of iterations is not known beforehand.

Basic syntax:

```
while condition:  
    # code to be executed
```

Example:

```
count = 0  
while count < 5:  
    print(count)  
    count += 1
```

0
1
2
3
4

Loop Control Statements

Python provides several statements to control the flow of loops:

1. `break`: Terminates the loop prematurely
2. `continue`: Skips the rest of the code inside the loop for the current iteration
3. `else`: Specifies a block of code to be executed when the loop is exhausted

Example:

```
for num in range(10):  
    if num == 5:  
        break  
    print(num)  
else:  
    print("Loop completed normally")
```

0
1
2
3
4

Nested Loops

Loops can be nested inside other loops, allowing for more complex iterations:

```
for i in range(3):  
    for j in range(3):  
        print(f"({i}, {j})")
```

(0, 0)
(0, 1)
(0, 2)
(1, 0)
(1, 1)
(1, 2)
(2, 0)
(2, 1)
(2, 2)

List Comprehensions

Python offers a concise way to create lists based on existing lists, known as list comprehensions:

```
squares = [x**2 for x in range(10)]  
print(squares)
```

[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]

This is equivalent to:

```
squares = []  
for x in range(10):  
    squares.append(x**2)  
print(squares)
```

[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]