For Loop in Python

====================

A `for` loop in Python is a control structure that allows you to execute a block of code for each item in an iterable (such as a list, tuple, or string).

Basic Syntax

The basic syntax of a `for` loop in Python is as follows:

```python
for variable in iterable:
# do something with the variable
```

Here, `variable` is the name given to the current item being processed in each iteration, and `iterable` is the sequence of items that will be processed.

Example

Let's consider an example where we want to print out the numbers from 1 to 5:

```python
fruits = ['apple', 'banana', 'cherry']
```

```
for fruit in fruits:
print(fruit)
```

This will output:

```
apple
banana
cherry
```

In this example, `fruit` is the variable that takes on the value of each item in the `fruits` list in each iteration.

Multiple Iterables

You can also use multiple iterables by separating them with commas. For example:

```python
numbers = [1, 2, 3]
letters = ['a', 'b', 'c']

for num, letter in zip(numbers, letters):
print(f"Number: {num}, Letter: {letter}")
```

This will output:

```
Number: 1, Letter: a
Number: 2, Letter: b
Number: 3, Letter: c
```

Looping Through Dictionaries

You can also use `for` loops to iterate over the key-value pairs of dictionaries. For example:

```python
person = {'name': 'John', 'age': 30}

for key, value in person.items():
print(f"{key}: {value}")
```

This will output:

```
name: John
age: 30
```

Breaking and Continuing

You can use the `break` statement to exit a loop prematurely, or the `continue` statement to skip to the next iteration.

For example:

```python
for i in range(10):
if i == 5:
break
print(i)
```

This will output:

```
0
1
2
3
4
```

And:

```python
for i in range(10):
if i == 5:
continue
print(i)
```

This will output:

```
0
```

1

2

3

4

6

7

8

9

```
```