Python for else

Summary: in this tutorial, you'll learn about the Python `for else` statement and how to use it effectively.

Introduction to the Python for else statement

In Python, the `for` (https://www.pythontutorial.net/python-basics/python-for-range/) statement can have an optional `else` clause, which you may not be familiar with especially if you're coming from other languages such as Java or C#.

The following shows the syntax of the `for` statement with the `else` clause:

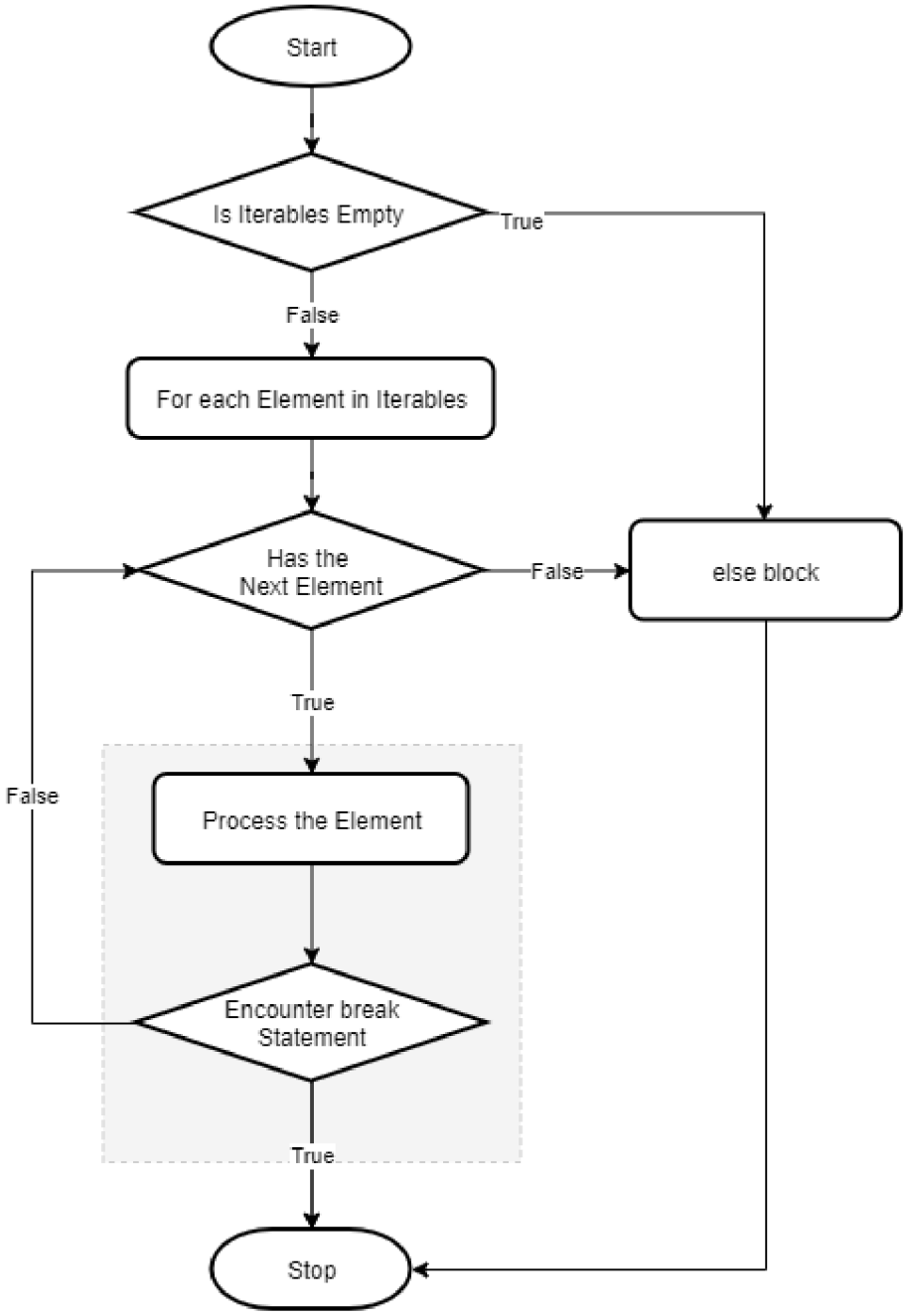
```
for item in iterables:
    # process item
else:
    # statement
```

In this syntax, the `else` clause will execute only if the loop runs **normally**. In other words, the `else` clause won't execute if the loop encounters a `break` (https://www.pythontutorial.net/python-basics/python-

`break/)` statement.

In addition, the `else` clause also executes when the iterables object has no item.

The following flowchart illustrates the for...else statement:



The `else` clause is quite useful in some cases if you know how to apply it effectively.

Python for else statement example

Suppose that you have a list (https://www.pythontutorial.net/python-basics/python-list/) of people, where each person is a dictionary (https://www.pythontutorial.net/python-basics/python-dictionary/) that consists of `name` and `age` like this:

```
people = [{'name': 'John', 'age': 25},
        {'name': 'Jane', 'age': 22},
        {'name': 'Peter', 'age': 30},
        {'name': 'Jenifer', 'age': 28}]
```

And you want to search for a person by name.

If the list contains the person, you want to display the information of that person. Otherwise, you want to show a message saying that the name is not found.

To do it, you may come up with a program like this:

```
people = [{'name': 'John', 'age': 25},
        {'name': 'Jane', 'age': 22},
        {'name': 'Peter', 'age': 30},
        {'name': 'Jenifer', 'age': 28}]

name = input('Enter a name:')

found = False
for person in people:
    if person['name'] == name:
        found = True
        print(person)
        break

if not found:
    print(f'{name} not found!')
```

How it works:

- First, prompt for a name by using the `input()` function.
- Then, set a flag (`found`) to `False` . If the input name matches with a person on the list, set its value to `True` , show the person's information and exit the loop by using the `break` statement.
- Finally, check the `found` flag and show a message.

The following runs a program with the name `Peter` and `Maria` :

1st run:

```
Enter a name:Peter
{'name': 'Peter', 'age': 30}
```

2nd run:

```
Enter a name:Maria
Maria not found!
```

It works perfectly fine.

However, if you use the `for else` statement, the program will be much shorter.

The following shows the new version of the program that uses the `for else` statement:

```
people = [{'name': 'John', 'age': 25},
        {'name': 'Jane', 'age': 22},
        {'name': 'Peter', 'age': 30},
        {'name': 'Jenifer', 'age': 28}]

name = input('Enter a name:')

for person in people:
    if person['name'] == name:
        print(person)
        break
```

```
else:
    print(f'{name} not found!')
```

By using the `for else` statement, the program doesn't need to use a `flag` and an `if` (https://www.pythontutorial.net/python-basics/python-if/) statement after the loop.

In this new program, if the input name matches a person on the list, it'll show the person's information and exit the loop by using the `break` statement.

When the loop encounters the `break` statement, the `else` clause won't execute.

Summary

- Use Python `for else` statement to execute a code block if the loop doesn't encounter a `break` statement or if the iterables object has no item.