Python enum auto

Summary: in this tutorial, you'll learn about the enum auto() function to generate unique values for enumeration members.

Introduction to the enum auto() function

The following example defines an enumeration (https://www.pythontutorial.net/python-oop/python-enumeration/) with three members whose values are 1, 2, and 3:

```
from enum import Enum


class State(Enum):
    PENDING = 1
    FULFILLED = 2
    REJECTED = 3
```

In this example, we manually assign integer values to the members of the enumeration.

To make it more convenient, Python 3.6 introduced the `auto()` helper class in the `enum` module, which automatically generates unique values for the enumeration members. For example:

```
from enum import Enum, auto


class State(Enum):
    PENDING = auto()
    FULFILLED = auto()
    REJECTED = auto()

    def __str__(self):
        return f'{self.name(self.value)}'
```

How it works.

- First, import the `Enum` and `auto` classes from the `enum` module.
- Second, call the `auto()` to generate a unique value for each member of the `State` enumeration.

By default, the `auto()` class generates a sequence of integer numbers starting from 1.

The following shows the values of the `State` enumeration's members:

```
for state in State:
    print(state.name, state.value)
```

Output:

```
PENDING 1
FULFILLED 2
REJECTED 3
```

How enum() auto works

Technically, the `auto()` calls the `_generate_next_value_()` method to generate values for the members. Here's the syntax of the `_generate_next_value_()` method:

```
_generate_next_value_(name, start, count, last_values)
```

The `_generate_next_value_()` has the following parameters:

- `name` is the member's name
- `start` is the starting value of the enum members.
- `count` is the number of enum members, including aliases (https://www.pythontutorial.net/python-oop/python-enum-unique/) , that have been created.
- `last_values` is a list of all preceding values used for the enum members.

By default, the `_generate_next_value_()` generates the next number in a sequence of integers starting from one. However, Python may change this logic in the future.

It's possible to override (https://www.pythontutorial.net/python-oop/python-overriding-method/) the `_generate_next_value_()` method to add a custom logic that generates unique values. If so, you need to place the `_generate_next_value_()` method before defining all the members.

The following shows how to override the `_generate_next_value_()` method to generate values for members by using their names:

```
from enum import Enum, auto


class State(Enum):
    def _generate_next_value_(name, start, count, last_values):
        return name.lower()

    PENDING = auto()
    FULFILLED = auto()
    REJECTED = auto()


for state in State:
    print(state.name, state.value)
```

Output:

```
PENDING pending
FULFILLED fulfilled
REJECTED rejected
```

Summary

- Use enum `auto()` class to generate unique values for enumeration members.