Python - Referências Fracas

Python usa mecanismo de contagem de referência ao implementar a política de coleta de lixo. Sempre que um objeto na memória é referenciado, a contagem é incrementada em um. Por outro lado, quando a referência é removida, a contagem é decrementada em 1. Se o coletor de lixo em execução em segundo plano encontrar algum objeto com contagem 0, ele será removido e a memória ocupada será recuperada.

Referência fraca é uma referência que não protege o objeto contra a coleta de lixo. É importante quando você precisa implementar caches para objetos grandes, bem como em uma situação onde é desejada a redução do Pain de referências circulares.

Para criar referências fracas, Python nos forneceu um módulo chamado fracoref.

A classe ref neste módulo gerencia a referência fraca a um objeto. Quando chamado, recupera o objeto original.

Para criar uma referência fraca -

```
weakref.ref(class())
```

Exemplo

```
import weakref
class Myclass:
   def __del__(self):
      print('(Deleting {})'.format(self))
obj = Myclass()
r = weakref.ref(obj)

print('object:', obj)
print('reference:', r)
print('call r():', r())

print('deleting obj')
del obj
print('r():', r())
```

Chamar o objeto de referência após excluir o referente retorna Nenhum.

Ele produzirá a seguinte **saída** -

```
object: <__main__.Myclass object at 0x00000209D7173290>
reference: <weakref at 0x00000209D7175940; to 'Myclass' at
```

```
0x00000209D7173290>
call r(): <__main__.Myclass object at 0x00000209D7173290>
deleting obj
(Deleting <__main__.Myclass object at 0x00000209D7173290>)
r(): None
```

A função de retorno de chamada

O construtor da classe ref possui um parâmetro opcional chamado função de retorno de chamada, que é chamado quando o objeto referido é excluído.

```
import weakref
class Myclass:
   def __del__(self):
      print('(Deleting {})'.format(self))
def mycallback(rfr):
   """called when referenced object is deleted"""
   print('calling ({})'.format(rfr))
obj = Myclass()
r = weakref.ref(obj, mycallback)

print('object:', obj)
print('reference:', r)
print('call r():', r())

print('deleting obj')
del obj
print('r():', r())
```

Ele produzirá a seguinte **saída** -

```
object: <__main__.Myclass object at 0x000002A0499D3590>
reference: <weakref at 0x000002A0499D59E0; to 'Myclass' at
0x000002A0499D3590>
call r(): <__main__.Myclass object at 0x000002A0499D3590>
deleting obj
(Deleting <__main__.Myclass object at 0x000002A0499D3590>)
calling (<weakref at 0x000002A0499D59E0; dead>)
r(): None
```

Finalizando Objetos

O módulo **fracoref** fornece classe de finalização. Seu objeto é chamado quando o coletor de lixo coleta o objeto. O objeto sobrevive até que o objeto de referência seja chamado.

```
import weakref
class Myclass:
   def __del__(self):
      print('(Deleting {})'.format(self))

def finalizer(*args):
   print('Finalizer{!r})'.format(args))

obj = Myclass()
r = weakref.finalize(obj, finalizer, "Call to finalizer")

print('object:', obj)
print('reference:', r)
print('call r():', r())

print('deleting obj')
del obj
print('r():', r())
```

Ele produzirá a seguinte **saída** -

```
object: <__main__.Myclass object at 0x0000021015103590>
reference: <finalize object at 0x21014eabe80; for 'Myclass' at
0x21015103590>
Finalizer('Call to finalizer',))
call r(): None
deleting obj
(Deleting <__main__.Myclass object at 0x0000021015103590>)
r(): None
```

O módulo fracoref fornece classes WeakKeyDictionary e WeakValueDictionary. Eles não mantêm os objetos vivos conforme aparecem nos objetos de mapeamento. Eles são mais apropriados para criar um cache de vários objetos.

Dicionário WeakKey

Classe de mapeamento que faz referência fraca às chaves. As entradas do dicionário serão descartadas quando não houver mais uma referência forte à chave.

Uma instância da classe WeakKeyDictionary é criada com um dicionário existente ou sem qualquer argumento. A funcionalidade é a mesma de um dicionário normal para adicionar e remover entradas de mapeamento a ele.

No código fornecido a seguir, três instâncias de Person são criadas. Em seguida, ele cria uma instância de WeakKeyDictionary com um dicionário onde a chave é a instância de Person e o valor é o nome da Pessoa.

Chamamos o método keyrefs() para recuperar referências fracas. Quando a referência a Peron1 é excluída, as chaves do dicionário são impressas novamente. Uma nova instância de Person é adicionada a um dicionário com chaves com referência fraca. Por fim, estamos imprimindo as chaves do dicionário novamente.

Exemplo

```
import weakref

class Person:
   def __init__(self, person_id, name, age):
      self.emp_id = person_id
      self.name = name
      self.age = age

   def __repr__(self):
      return "{} : {} : {}".format(self.person_id, self.name, self.age)
Person1 = Person(101, "Jeevan", 30)
Person2 = Person(102, "Ramanna", 35)
Person3 = Person(103, "Simran", 28)
weak_dict = weakref.WeakKeyDictionary({Person1: Person1.name, Person2: Person2.name,
print("Weak Key Dictionary : {}\n".format(weak_dict.data))
print("Dictionary Keys : {}\n".format([key().name for key in weak_dict.keyrefs()]))
del Person1
print("Dictionary Keys : {}\n".format([key().name for key in weak_dict.keyrefs()]))
Person4 = Person(104, "Partho", 32)
weak_dict.update({Person4: Person4.name})

print("Dictionary Keys : {}\n".format([key().name for key in weak_dict.keyrefs()]))
```

Ele produzirá a seguinte **saída** -

```
Weak Key Dictionary : {<weakref at 0x7f542b6d4180; to 'Person' at 0x7f542b8bbfd0>: 'Jeevan', <

Dictionary Keys : ['Jeevan', 'Ramanna', 'Simran']

Dictionary Keys : ['Ramanna', 'Simran']

Dictionary Keys : ['Ramanna', 'Simran', 'Partho']
```

Dicionário de Valor Fraco

Classe de mapeamento que faz referência fraca a valores. As entradas no dicionário serão descartadas quando não existir mais nenhuma referência forte ao valor.

Demonstraremos como criar um dicionário com valores fracamente referenciados usando WeakValueDictionary.

O código é semelhante ao exemplo anterior, mas desta vez estamos usando o nome da pessoa como chave e a instância da pessoa como valores. Estamos usando o método valuerefs() para recuperar valores fracamente referenciados do dicionário.

Exemplo

```
import weakref

class Person:
   def __init__(self, person_id, name, age):
      self.emp_id = person_id
      self.name = name
      self.age = age

   def __repr__(self):
      return "{} : {} : {}".format(self.person_id, self.name, self.age)

Person1 = Person(101, "Jeevan", 30)
Person2 = Person(102, "Ramanna", 35)
Person3 = Person(103, "Simran", 28)

weak_dict = weakref.WeakValueDictionary({Person1.name:Person1, Person2.name:Person2,
print("Weak Value Dictionary : {}\n".format(weak_dict.data))
print("Dictionary Values : {}\n".format([value().name for value in weak_dict.valueref
del Person1
print("Dictionary Values : {}\n".format([value().name for value in weak_dict.valueref
Person4 = Person(104, "Partho", 32)
weak_dict.update({Person4.name: Person4})
print("Dictionary Values : {}\n".format([value().name for value in weak_dict.valueref
```

Ele produzirá a seguinte **saída** -

```
Weak Value Dictionary : {'Jeevan': <weakref at 0x7f3af9fe4180; to 'Person' at 0x7f3afa1c7fd0>, 'F

Dictionary Values : ['Jeevan', 'Ramanna', 'Simran']

Dictionary Values : ['Ramanna', 'Simran']
```

```
Dictionary Values : ['Ramanna', 'Simran', 'Partho']
```