
Subject: lambda functions in IDL-python bridge

Posted by [Helder Marchetto](#) on Fri, 04 Sep 2015 13:55:28 GMT

[View Forum Message](#) <> [Reply to Message](#)

Hi,

I've got the next problem, using a lambda function in the python sorted function.

Consider this easy python example:

```
d0 = {'sorted': 1, 'unsorted': 0}
d1 = {'sorted': 5, 'unsorted': 1}
d2 = {'sorted': 3, 'unsorted': 2}
d3 = {'sorted': 7, 'unsorted': 3}
d4 = {'sorted': 4, 'unsorted': 4}
l = [d0,d1,d2,d3,d4]
sorted(l, key = lambda x:x['sorted'])
```

The result would be:

```
[{'sorted': 1, 'unsorted': 0},
 {'sorted': 3, 'unsorted': 2},
 {'sorted': 4, 'unsorted': 4},
 {'sorted': 5, 'unsorted': 1},
 {'sorted': 7, 'unsorted': 3}]
```

Now if I would want to do this in IDL, things would get difficult. Suppose I get a dictionary (or a simple object with a property...) as a result of an operation and I want to sort this result according to a specific property... how do I do that in IDL?

The synthax:

```
matches = Python.sorted(matches, key = lambda(x: x.distance))
```

will give me a compilation error.

Thanks,
Helder

PS: I found a different solution to this problem using IDL. I import all properties in an idl variable, sort the variable and apply this to the list. A python solution would be though nice, just to know how to deal with lambda functions.

Subject: Re: lambda functions in IDL-python bridge

Posted by [chris_torrence@NOSPAM](#) on Tue, 08 Sep 2015 15:34:09 GMT

[View Forum Message](#) <> [Reply to Message](#)

On Friday, September 4, 2015 at 7:55:30 AM UTC-6, Helder wrote:

> Hi,

> I've got the next problem, using a lambda function in the python sorted function.

```

>
> Consider this easy python example:
>
> d0 = {'sorted': 1, 'unsorted': 0}
> d1 = {'sorted': 5, 'unsorted': 1}
> d2 = {'sorted': 3, 'unsorted': 2}
> d3 = {'sorted': 7, 'unsorted': 3}
> d4 = {'sorted': 4, 'unsorted': 4}
> l = [d0,d1,d2,d3,d4]
> sorted(l, key = lambda x:x['sorted'])
>
> The result would be:
> [{'sorted': 1, 'unsorted': 0},
>  {'sorted': 3, 'unsorted': 2},
>  {'sorted': 4, 'unsorted': 4},
>  {'sorted': 5, 'unsorted': 1},
>  {'sorted': 7, 'unsorted': 3}]
>
> Now if I would want to do this in IDL, things would get difficult. Suppose I get a dictionary (or a
simple object with a property...) as a result of an operation and I want to sort this result according
to a specific property... how do I do that in IDL?
>
> The synthax:
> matches = Python.sorted(matches, key = lambda(x: x.distance))
>
> will give me a compilation error.
>
> Thanks,
> Helder
>
> PS: I found a different solution to this problem using IDL. I import all properties in an idl
variable, sort the variable and apply this to the list. A python solution would be though nice, just to
know how to deal with lambda functions.

```

Hi Helder,

Unfortunately, IDL can't translate its Lambda functions into Python lambda functions. If you wanted to have Python do this, you could transfer the variable over to Python, and then execute a Python command. Like this:

```

IDL> d0 = hash('sorted', 1, 'unsorted', -1)
IDL> d1 = hash('sorted', 5, 'unsorted', 1)
IDL> d2 = hash('sorted', 3, 'unsorted', 2)
IDL> d3 = hash('sorted', 7, 'unsorted', 3)
IDL> d4 = hash('sorted', 4, 'unsorted', 4)
IDL> lst = list(d0,d1,d2,d3,d4)
IDL> Python.l = lst
IDL> >>>result = sorted(l, key = lambda x:x['sorted'])

```

```
IDL> result = Python.result
IDL> print, result, /implied
```

Now, if you wanted to do this purely in IDL, you could use List::Sort with a Lambda compare function, and the new .Compare method, which exists on IDL variables:

```
d0 = {sorted: 1, unsorted: 0}
d1 = {sorted: 5, unsorted: 1}
d2 = {sorted: 3, unsorted: 2}
d3 = {sorted: 7, unsorted: 3}
d4 = {sorted: 4, unsorted: 4}
lst = List(d0,d1,d2,d3,d4)
result = lst.Sort(COMPARE=Lambda(a,b:(a.sorted).Compare(b.sorted)))
print, result, /implied
```

Here's an example of how the .Compare method works:

```
IDL> print, (1).compare(2)
-1
IDL> print, (1).compare(1)
0
IDL> print, (1).compare(0)
1
```

Cheers,
Chris

Subject: Re: lambda functions in IDL-python bridge
Posted by [Helder Marchetto](#) on Wed, 09 Sep 2015 07:27:37 GMT
[View Forum Message](#) <> [Reply to Message](#)

On Tuesday, September 8, 2015 at 5:34:11 PM UTC+2, Chris Torrence wrote:

> On Friday, September 4, 2015 at 7:55:30 AM UTC-6, Helder wrote:

```
>> Hi,
>> I've got the next problem, using a lambda function in the python sorted function.
>>
>> Consider this easy python example:
>>
>> d0 = {'sorted': 1, 'unsorted': 0}
>> d1 = {'sorted': 5, 'unsorted': 1}
>> d2 = {'sorted': 3, 'unsorted': 2}
>> d3 = {'sorted': 7, 'unsorted': 3}
>> d4 = {'sorted': 4, 'unsorted': 4}
>> l = [d0,d1,d2,d3,d4]
>> sorted(l, key = lambda x:x['sorted'])
>>
>> The result would be:
>> [{'sorted': 1, 'unsorted': 0},
```

```

>> {'sorted': 3, 'unsorted': 2},
>> {'sorted': 4, 'unsorted': 4},
>> {'sorted': 5, 'unsorted': 1},
>> {'sorted': 7, 'unsorted': 3}}
>>
>> Now if I would want to do this in IDL, things would get difficult. Suppose I get a dictionary (or a
>> simple object with a property...) as a result of an operation and I want to sort this result according
>> to a specific property... how do I do that in IDL?
>>
>> The synthax:
>> matches = Python.sorted(matches, key = lambda(x: x.distance))
>>
>> will give me a compilation error.
>>
>> Thanks,
>> Helder
>>
>> PS: I found a different solution to this problem using IDL. I import all properties in an idl
>> variable, sort the variable and apply this to the list. A python solution would be though nice, just to
>> know how to deal with lambda functions.
>
> Hi Helder,
>
> Unfortunately, IDL can't translate its Lambda functions into Python lambda functions. If you
> wanted to have Python do this, you could transfer the variable over to Python, and then execute a
> Python command. Like this:
>
> IDL> d0 = hash('sorted', 1, 'unsorted', -1)
> IDL> d1 = hash('sorted', 5, 'unsorted', 1)
> IDL> d2 = hash('sorted', 3, 'unsorted', 2)
> IDL> d3 = hash('sorted', 7, 'unsorted', 3)
> IDL> d4 = hash('sorted', 4, 'unsorted', 4)
> IDL> lst = list(d0,d1,d2,d3,d4)
> IDL> Python.l = lst
> IDL> >>>result = sorted(l, key = lambda x:x['sorted'])
> IDL> result = Python.result
> IDL> print, result, /implied
>
> Now, if you wanted to do this purely in IDL, you could use List::Sort with a Lambda compare
> function, and the new .Compare method, which exists on IDL variables:
>
> d0 = {sorted: 1, unsorted: 0}
> d1 = {sorted: 5, unsorted: 1}
> d2 = {sorted: 3, unsorted: 2}
> d3 = {sorted: 7, unsorted: 3}
> d4 = {sorted: 4, unsorted: 4}
> lst = List(d0,d1,d2,d3,d4)
> result = lst.Sort(COMPARE=Lambda(a,b:(a.sorted).Compare(b.sorted)))

```

```
> print, result, /implied
>
> Here's an example of how the .Compare method works:
> IDL> print, (1).compare(2)
>      -1
> IDL> print, (1).compare(1)
>       0
> IDL> print, (1).compare(0)
>       1
>
> Cheers,
> Chris
```

Hi Chris,
thanks for the info. I'm fighting to understand how to interface python and IDL. Your examples are very nice and hope that more of such will come (that's a suggestion...).

Your IDL solution looks very exotic (=cool).
Cheers,
Helder
