Python from MATLAB

Daniel Driver

E177-Advanced Matlab

April 21, 2015
First Impressions

- Not available until 2014b
- More functionality in 2015a (according to the manual)
  - Index a List or Tuple with \{\}
- Pretty cumbersome at the moment
- No terrible though if you know python well
- Sometimes requires calling operator function instead of symbol
  - getitem instead of using “a=something[3]”
  - setitem instead of “something[3]=a”
- Binary operations all seem to work though
  - e.g. +,-,*,>
vehicleClass

class VehicleClass():
def __init__(self,*argv):# constructor
  import numpy as np
  self.Velocity=np.array([0,0],dtype=float)
  try:
    self.NumPassengers=argv[1]
  except:
    self.NumPassengers=1
  self.StorageVolume=0.0
  self.Color=argv[0]
def __add__(V1,V2): #overloads the '+' symbol
  a.Velocity=V1.Velocity+V2.Velocity
  return a
def __repr__(self): #Like Matlab Display Function
  OutputString=""" Vehicle with:
  Velocity {0},
  Storage Volume {1},
  Carries {2} Passangers
  and is the Color {3}""".format(self.Velocity,
                                  self.StorageVolume,
                                  self.NumPassengers,
                                  self.Color)
  return OutputString
def accelerate(self,deltaV):
  self.Velocity+=deltaV
Using a Python Class

**Python Usage**

```python
import PyClass import numpy as np
V1=PyClass.VehicleClass('Red')
V2=PyClass.VehicleClass('Yellow',5)

V1.accelerate(np.array([1,2]))
V2.accelerate(np.array([3,4]))

V3=V1+V2
V3.accelerate(np.array([5,1.3]))
print(V3)

multout=V3.Velocity*V1.Velocity
print('multout={0}'.format(multout))
```

- Do not do “import PyClass”
  - `py.PyClass` automatically imports PyClass
- Replace “from x import y”
  - “import x.y” in MATLAB

---

**Matlab Usage**

```matlab
V1=py.PyClass.VehicleClass('Red');
V2=py.PyClass.VehicleClass('Yellow',5);
V1.accelerate(py.numpy.array([1,2]));
V2.accelerate(py.numpy.array([3,4]));

V3=V1+V2;
V3.accelerate(py.numpy.array([5,1.3]));
py.print(V3)

display('multout=')
multout=V3.Velocity * V1.Velocity
```

- Pretty Clean MATLAB syntax
- “py” to first call a python Class
- after that mostly works
  - Only array inputs a little weird
Trying to work with Numpy

- Have to use pure python operators to index
  - getitem
  - setitem
- Still able to do everything with native python commands

MATLAB Usage

```matlab
% used pure python operators to work with numpy array
test = py.numpy.random.random(py.tuple([5, 5]))
py.operatorgetitem(test, py.tuple([3, 3]))
py.operatorsetitem(test, py.tuple([0, 1]), 100.0)
% will return python values
value = py.operatorgetitem(test, py.tuple([0, 1]))
% can divide like normal value / 3.3
display('display test')
display(test)
display('vs py.print method')
py.print(test)
display('Transpose of T')
test.T
```
Basic Indexing with {}

Work with
- Dictionary
- Tuple
- List

MATLAB Usage

```matlab
% put things in a dictionary
customers = py.dict
customers{'Smith'} = int32(2112);
customers{'Anderson'} = int32(3010);
customers{'Audrey'} = int32(4444);
customers{'Megan'} = int32(5000);
acct = customers{'Anderson'}

% make and loop over a list
li = py.list({1,2,3,4});
for n = li
disp(n{1})
end
```
MATLAB Usage

```matlab
%indexing into string (1 indexed)
pstr = py.str('myfile');
pstr(1)

%tuple slicing (1 indexed)
t = py.tuple({'a','bc',1,2,'def'});
t(1:2)

%list splicing matlab style
% start:step:stop
%also 1 indexed
li = py.list({'a','bc',1,2,'def'});
li(1:2:end)

%can also get end
%get value
li{end}
%get python object of value
li(end)
```

{} syntax
- seems to actually get value in matlab form
- only can get one value

() syntax
- makes another python object
- can perform 1-d slicing
Limitations-Indexing

You can access data in Python container objects, like lists and dictionaries, with index values, similar to referencing an element in a MATLAB matrix. There are, however, ways to index into matrices which are not supported for these Python types.

<table>
<thead>
<tr>
<th>Indexing Features Not Supported in MATLAB</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Use of square brackets, [ ].</td>
<td></td>
</tr>
<tr>
<td>Indexing into a container type that does not inherit from collections.Sequence or collections.Mapping.</td>
<td></td>
</tr>
<tr>
<td>Logical indexing.</td>
<td></td>
</tr>
<tr>
<td>Accessing data in a container with an arbitrary array of indices. An index must be of the form start:step:stop.</td>
<td></td>
</tr>
<tr>
<td>Comma-separated lists.</td>
<td></td>
</tr>
<tr>
<td>numel function does not return number of array elements. Returns 1.</td>
<td></td>
</tr>
</tbody>
</table>

**Figure**: Current limitations of indexing Python Objects in MATLAB as of 2015a
**Limitations-General**

<table>
<thead>
<tr>
<th>Features Not Supported in MATLAB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Editing and reloading a Python module in the same MATLAB session. To use an updated module, restart MATLAB.</td>
</tr>
<tr>
<td>Closing the Python interpreter while running MATLAB.</td>
</tr>
<tr>
<td>Saving (serializing) Python objects into a MAT-file.</td>
</tr>
<tr>
<td>Interactive Python help (calling <code>py.help</code> without input arguments).</td>
</tr>
<tr>
<td><code>py.input</code> and <code>py.raw_input</code> (version 2.7).</td>
</tr>
<tr>
<td>Accessing static properties of a Python class.</td>
</tr>
<tr>
<td>MATLAB <code>isa</code> function does not recognize virtual inheritance.</td>
</tr>
<tr>
<td>MATLAB class inheritance from a Python class.</td>
</tr>
<tr>
<td>Overloaded attribute access.</td>
</tr>
<tr>
<td>Nested Python classes.</td>
</tr>
<tr>
<td>Modules that start MATLAB in a separate process, for example, the <code>multiprocessing</code> module.</td>
</tr>
<tr>
<td>Modules that read <code>sys.argv</code>, the command-line arguments passed to a Python script, for example, <code>Tkinter</code>.</td>
</tr>
<tr>
<td>Dynamically generated Python classes, for example, <code>collections.namedtuple</code>.</td>
</tr>
<tr>
<td>Dynamically attaching new object attributes. Instead, use <code>py.setattr</code>.</td>
</tr>
<tr>
<td>Class names or other identifiers starting with an underscore <code>_</code> character. Instead, use the Python <code>py.getattr</code> and <code>py.setattr</code> functions.</td>
</tr>
</tbody>
</table>

**Figure:** General Limitations as of MATLAB 2015a