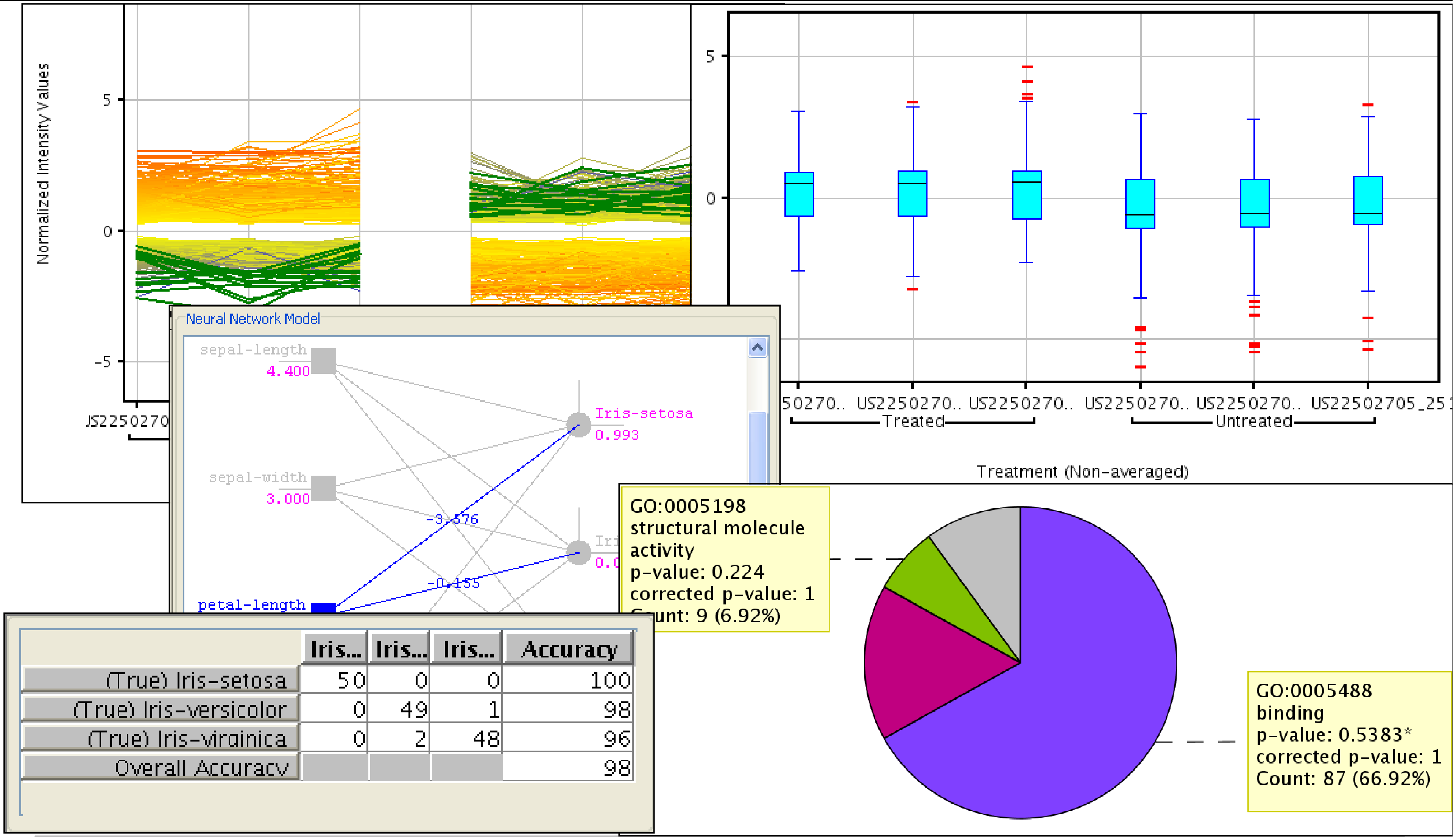
python @ Strand

Strand Life Sciences
Algorithms for Life

Overview

- Strand's avadis^{TM} platform
- Used in several verticals:
  - Microarray expression (GeneSpring)
  - Chemical structure descriptor (Sarchitect)
  - Next gen sequencing (faNGS)
  - stock market, semiconductor (potential)
- Data analysis and visualization:
  - Import tabular data
  - Perform visualizations and preprocessing
  - Execute analysis algorithms
  - Visualize results leading to discovery

Examples

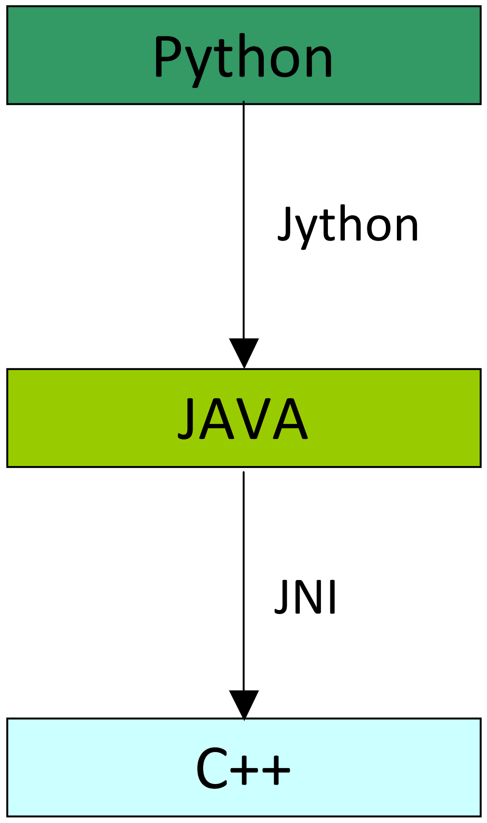


GO:0005198
structural molecule activity
p-value: 0.224
corrected p-value: 1
Count: 9 (6.92%)

GO:0005488
binding
p-value: 0.5383*
corrected p-value: 1
Count: 87 (66.92%)

	Iris...	Iris...	Iris...	Accuracy
(True) Iris-setosa	50	0	0	100
(True) Iris-versicolor	0	49	1	98
(True) Iris-virginica	0	2	48	96
Overall Accuracy				98

Architecture



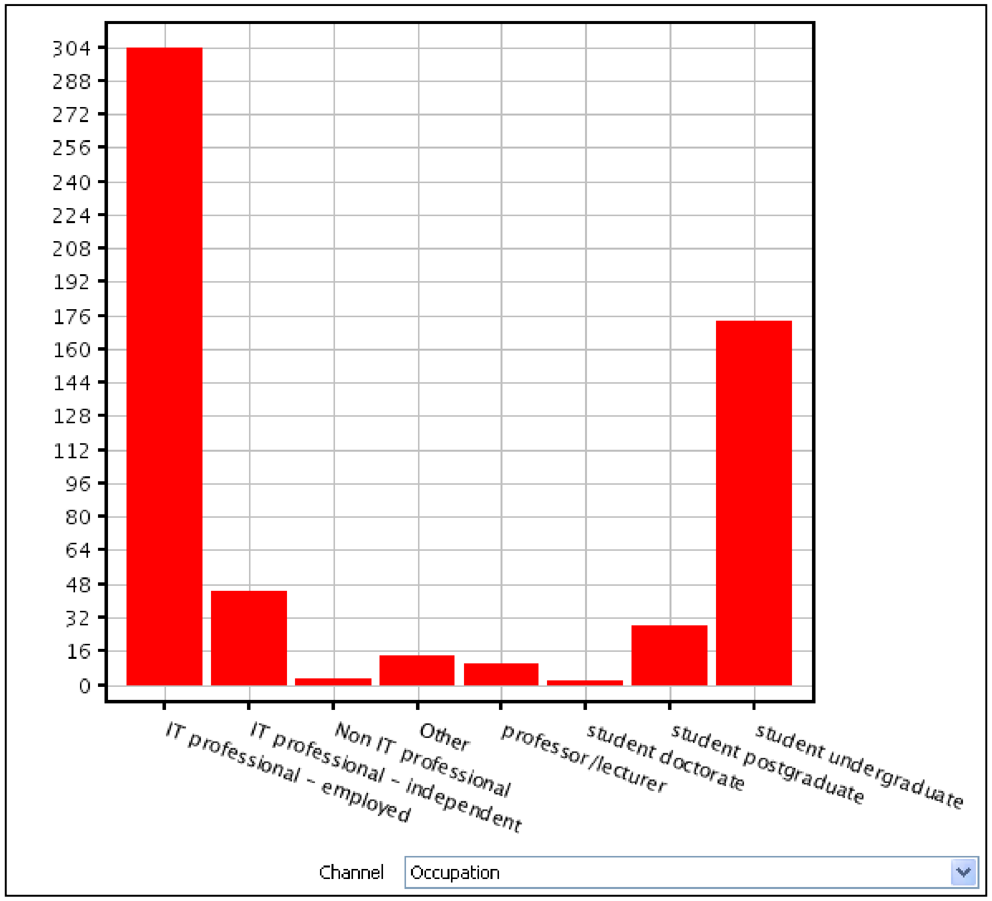
- Python:
  - Rapid development
  - Mix and match features
  - Fast debugging
- JAVA:
  - Core pluggable framework
  - User interface
  - Several algorithms
- C++:
  - Core/Legacy algorithms

avadis in action

```
# generic python code …
url = “http://in.pycon.org/2009/delegates/”
# python code to download and parse URL
# say using sgmllib.py
# to get lists of ids, names, occupation, city, etc.
tableData = extractTableData (url)

# avadis code starts here …
# create the dataset
from script.dataset import createStringColumn, createDataset
columns = []
for (name, data) in tableData:
    columns.append (createStringColumn (name, data)
d = script.dataset.createDataset (“delegates”, columns)

# launch a view on the dataset
view = script.view.Histogram (dataset=d, xLabelOrientation=“Slanted”)
view.show()
```

Strand Life Sciences
Algorithms for Life

avadis in action



Channel: Occupation

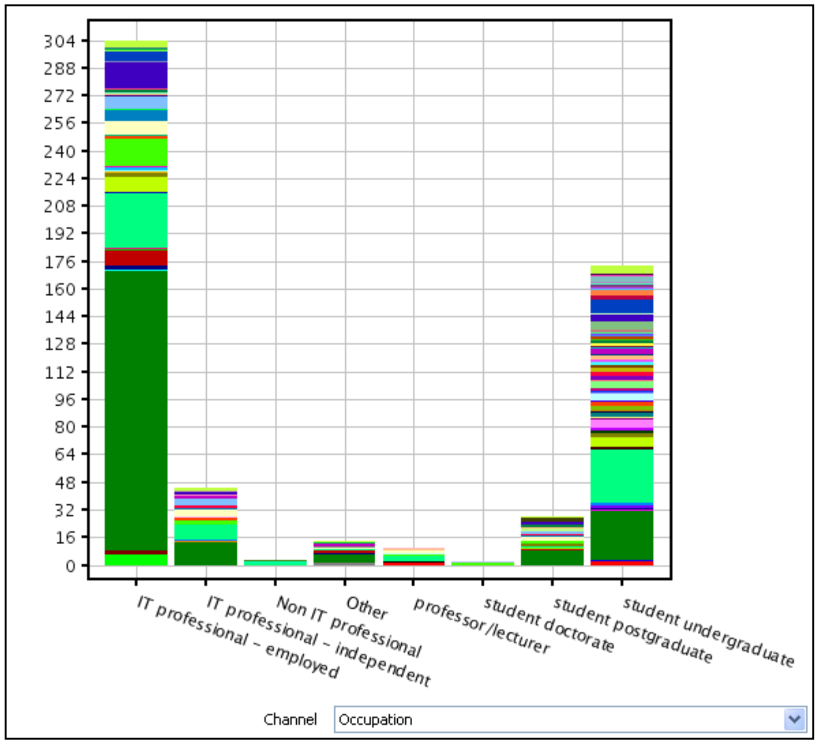
But that's already available at http://in.pycon.org/2009/statistics/
So, lets do something a little more interesting.

Strand Life Sciences

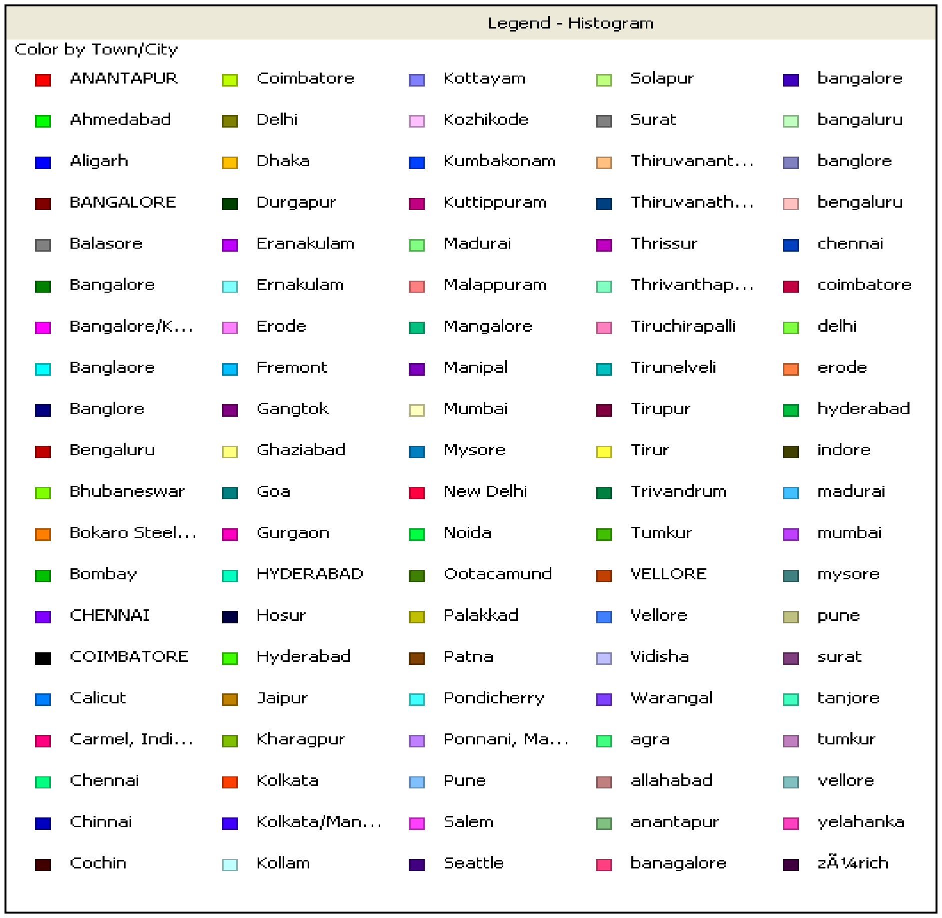
Algorithms for Life

avadis in action

```
view = script.view.Histogram (dataset=d, xLabelOrientation=“Slanted”)
view.colorBy.columnIndex = 4
view.show()
```



Channel: Occupation



Legend - Histogram

Color by Town/City

🟥 ANANTAPUR	🟩 Coimbatore	🟦 Kottayam	🟩 Solapur	🟪 bangalore
🟩 Ahmedabad	🟫 Delhi	⬜ Kozhikode	⬛ Surat	🟩 bangaluru
🟦 Aligarh	🟧 Dhaka	🟦 Kumbakonam	🟧 Thiruvanant...	🟦 banglore
🟥 BANGALORE	🟩 Durgapur	🟥 Kuttippuram	🟦 Thiruvanath...	🟥 bengaluru
⬛ Balasore	🟪 Eranakulam	🟩 Madurai	🟪 Thrissur	🟦 chennai
🟩 Bangalore	🟦 Ernakulam	🟥 Malappuram	🟩 Thrivanthap...	🟥 coimbatore
🟪 Bangalore/K...	🟪 Erode	🟩 Mangalore	🟥 Tiruchirapalli	🟩 delhi
🟦 Banglaore	🟦 Fremont	🟪 Manipal	🟦 Tirunelveli	🟧 erode
🟦 Banglore	🟪 Gangtok	⬜ Mumbai	🟥 Tirupur	🟩 hyderabad
🟥 Bengaluru	🟨 Ghaziabad	🟦 Mysore	🟨 Tirur	🟫 indore
🟩 Bhubaneswar	🟦 Goa	🟥 New Delhi	🟩 Trivandrum	🟦 madurai
🟧 Bokaro Steel...	🟪 Gurgaon	🟩 Noida	🟩 Tumkur	🟪 mumbai
🟩 Bombay	🟦 HYDERABAD	🟩 Ootacamund	🟧 VELLORE	🟦 mysore
🟪 CHENNAI	⬛ Hosur	🟨 Palakkad	🟦 Vellore	🟨 pune
⬛ COIMBATORE	🟩 Hyderabad	🟫 Patna	⬜ Vidisha	🟪 surat
🟦 Calicut	🟧 Jaipur	🟦 Pondicherry	🟪 Warangal	🟩 tanjore
🟥 Carmel, Indi...	🟩 Kharagpur	🟪 Ponnani, Ma...	🟩 agra	🟪 tumkur
🟩 Chennai	🟧 Kolkata	🟦 Pune	🟫 allahabad	🟦 vellore
🟦 Chinnai	🟦 Kolkata/Man...	🟪 Salem	🟩 anantapur	🟪 yelahanka
🟫 Cochin	⬜ Kollam	🟪 Seattle	🟥 banagalore	🟪 zÃ¼rich

Strand Life Sciences
Algorithms for Life

The Law of Choices

- Give a man a **single** choice, and he will gladly take it.
- Give a man **two** choices, and he will be confused.
- Give a man **three** choices, and he will run to his wife.
- Give a man **multiple** choices, and he will be doomed.
- Give a man **infinite** choices, and YOU are doomed.

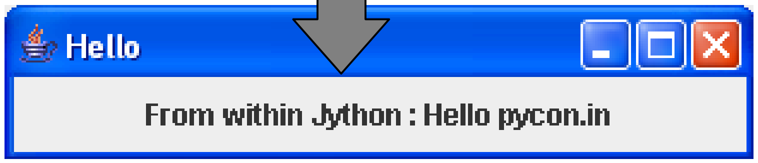
python - JAVA interface

- Jython : used for the python – JAVA interface.
- Jython **is** Python.
- Install Jython from http://www.jython.org/
- Use **`jython`** command line tool to execute Jython scripts.
- All JAVA classes are instantly accessible from within the Jython script.
- Additional JAVA classes are also accessible once the CLASSPATH variable is set.

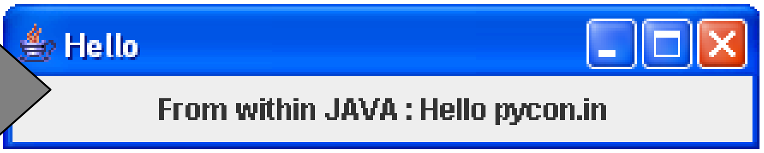
Example

```
import javax.swing.JFrame;
import javax.swing.JLabel;
import java.awt.Dimension;

public class Test {

  public static void main (String[] args)
  {
    JFrame f = new JFrame ("Hello");
    String s = "From within JAVA : Hello pycon.in";
    JLabel l = new JLabel (s, JLabel.CENTER);
    f.getContentPane().add (l);
    f.setSize (new Dimension (300, 50));
    f.setDefaultCloseOperation (f.EXIT_ON_CLOSE);
    f.setVisible (true);
  }
}

javac Test.java
java -cp . Test
```

```
moksha:jython2.5.1rc2$ ./jython
>>> from javax.swing import JFrame, JLabel
>>> f = JFrame ('Hello')
>>> t = 'From within Jython : Hello pycon.in'
>>> l = JLabel (t, JLabel.CENTER)
>>> f.contentPane.add (l)
>>> f.size = (300, 50)
>>> f.defaultCloseOperation = f.EXIT_ON_CLOSE
>>> f.visible = 1
```



From within Jython : Hello pycon.in



From within JAVA : Hello pycon.in

Strand Life Sciences
Algorithms for Life

Charming Jython

- Can nicely mix Python and JAVA code:

```
moksha:jython2.5.1rc2$ ./jython
>>> import random
>>> l = [random.randint (0, 100) for i in xrange (50)]
>>> from java.util import Collections
>>> Collections.sort (l)
```

- or, extend JAVA classes in Python:

```
moksha:jython2.5.1rc2$ ./jython
>>> from java.io import FileOutputStream
>>> class UppercaseFileOutputStream (FileOutputStream):
...     def write (self, text):
...         text = text.upper()
...         FileOutputStream.write (self, text)
...
>>> fos = UppercaseFileOutputStream ('out.txt')
>>> [fos.write ('This is line number ' + str(i) + '\n') for i in xrange(10)]
>>> fos.close()
```

- and more http://wiki.python.org/jython/LearningJython

Jython from JAVA

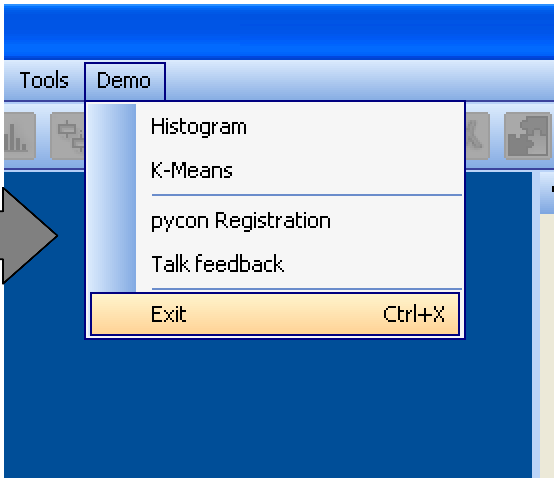
- PyInterpreter class (org.python.util package)
  - ➢ interp.exec (code)
  - ➢ interp.set (name, value)
  - ➢ interp.setOut (outstream)
  - ➢ interp.setErr (outstream)
- avadis has a thin layer of JAVA on top of Jython, which essentially does the above (JAVA6 has a better way of doing this – JAVA Scripting API).
- most of the user interaction with the application begins with python scripts.

Strand Life Sciences

Algorithms for Life

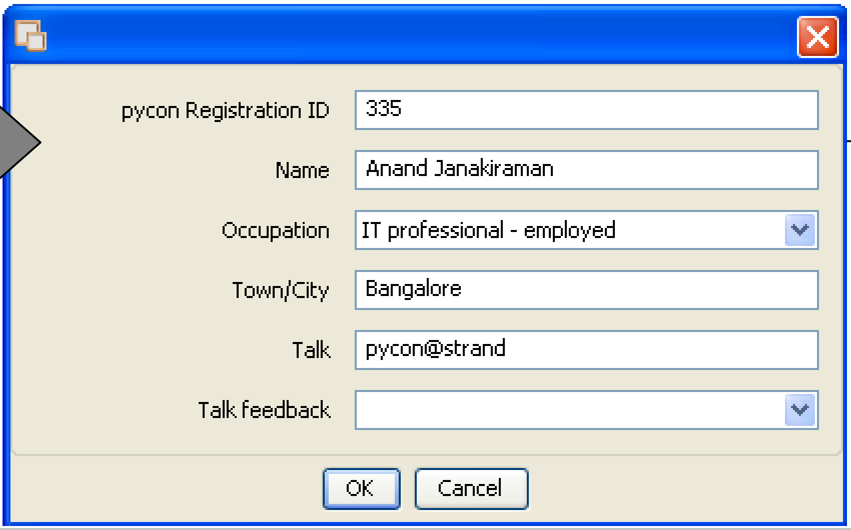
An example

```
<object type="spring.resource.menu.menuItem" version="1.0">
    <key>name</key>
    <string>K-Means</string>
    <key>mnemonic</key>
    <string>K</string>
    <key>accelerator</key>
    <string></string>
    <key>tooltip</key>
    <string>K-means</string>
    <key>action</key>
    <string>script.algorithm.KMeans().execute()</string>
</object>
<object type="spring.resource.menu.menuItem" version="1.0">
    <key>name</key>
    <string>Exit</string>
    <key>mnemonic</key>
    <string>X</string>
    <key>accelerator</key>
    <string>X</string>
    <key>tooltip</key>
    <string>Exit</string>
    <key>action</key>
    <string>java.lang.System.exit(0)</string>
</object>
```



Another example

```
from script.omega import createComponent, showDialog

c = script.project.getActiveDataset().getColumn ('Occupation')
occupations = [c.getCategoryValue(i) for i in xrange (c.getCategoryCount())]

c1 = createComponent (id='rid', type='int', description='pycon Registration ID')
c2 = createComponent (id='name', type='string', description='Name')
c3 = createComponent (id='occ', type='enum', description='Occupation', options=occupations)
c4 = createComponent (id='place', type='string', description='Town/City')
c5 = createComponent (id='talk', type='string', description='Talk', value='pycon@strand')
c6 = createComponent (id='fb', type='enum', description='Talk feedback', options=['', 'Stinks'])

c = createComponent (id='x', type='group', description='', components=[c1,c2,c3,c4,c5,c6])

v = showDialog (c)
print v
```



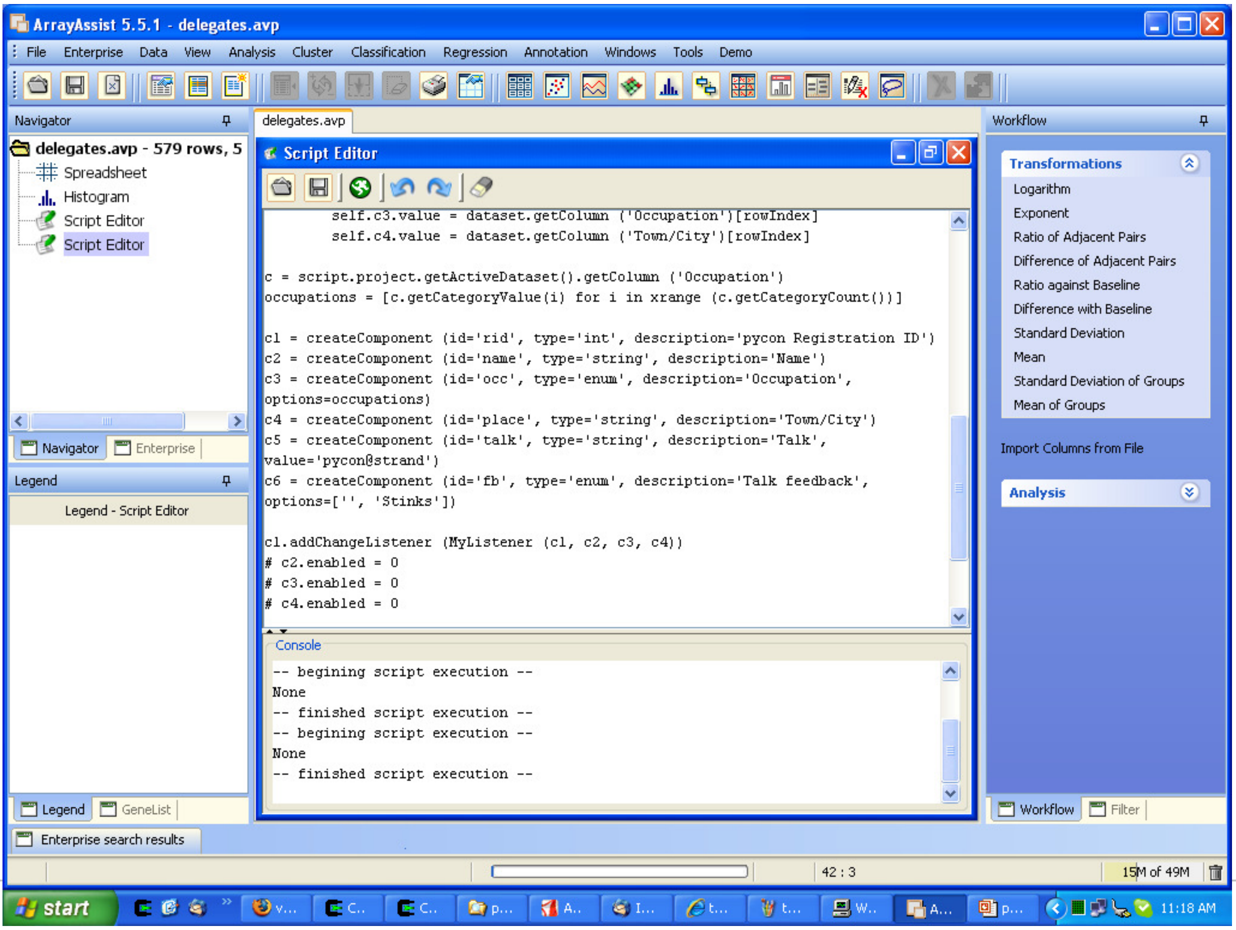
pycon Registration ID	335
Name	Anand Janakiraman
Occupation	IT professional - employed
Town/City	Bangalore
Talk	pycon@strand
Talk feedback	

Strand Life Sciences

Algorithms for Life

Debugging – differently

- The script editor and its beauty for debugging, and quickly trying out code.



```
        self.c3.value = dataset.getColumn ('Occupation')[rowIndex]
        self.c4.value = dataset.getColumn ('Town/City')[rowIndex]

c = script.project.getActiveDataset().getColumn ('Occupation')
occupations = [c.getCategoryValue(i) for i in xrange (c.getCategoryCount())]

c1 = createComponent (id='rid', type='int', description='pycon Registration ID')
c2 = createComponent (id='name', type='string', description='Name')
c3 = createComponent (id='occ', type='enum', description='Occupation',
options=occupations)
c4 = createComponent (id='place', type='string', description='Town/City')
c5 = createComponent (id='talk', type='string', description='Talk',
value='pycon@strand')
c6 = createComponent (id='fb', type='enum', description='Talk feedback',
options=['', 'Stinks'])

c1.addChangeListener (MyListener (c1, c2, c3, c4))
# c2.enabled = 0
# c3.enabled = 0
# c4.enabled = 0
```

Console

```
-- begining script execution --
None
-- finished script execution --
-- begining script execution --
None
-- finished script execution --
```

Strand Life Sciences

Algorithms for Life

Issues

- Started as a light wrapper, going on to become the heavyweight in the code base.
- Jython uses reflection internally => efficiency issues in making large number of JAVA calls from Jython – say within a for loop.
- Unlike JAVA, OOPS is not enforced => issues when programming in a larger software group.
- Compilation doesn’t capture JAVA compile errors, only syntax errors.

Strand Life Sciences
Algorithms for Life

Take homes

- The Law of Choices ☺
- A Scripting Engine for JAVA applications.
- Script Editor.
- Moderation ☺

Thank you