Writing Python Code for CARMEN Services

Python code can be easily wrapped as a service, but needs to be written as a command-line, with the required service input parameters as the command line parameters. The outputs must be printed to the screen/stdout, which will then be captured by the service management code.

Here is a simple string concatenator command line application. It takes two strings as input, concatenates them together, then prints the result string to the screen (i.e. the service output).

```
# Simple string concatenator
#!/usr/bin/env python
import sys

inStr0=sys.argv[1]
inStr1=sys.argv[2]
out = "<output>" + inStr0 + inStr1 + "</output>"
print out
```

Typically, we will be passing files in and out of the service. The following example takes three input parameters and outputs a reference to a filename. The three input parameters are an input filename, the name we require for the generated output filename, and a numeric value. The algorithm has been removed for clarity.

```
# Get parameters
infilename=sys.argv[1]
outfilename=sys.argv[2]
number=int(sys.argv[3])
#Open and read input file
infile=open(infilename)
incontents=infile.read()
infile.close()

# do stuff here
. . .

#Open and write output file
outfile=open(outfilename,  'w')
outfile.write(outputstring)
outfile.close()
print "<output>" + outfilename + "</output>"
```

If multiple outputs are required from the service, then they must be concatenated into a comma-seperated string. It is best to create a string and use that to build the comma-seperated output string, then print the string, i.e.

```
outputStr = "<output>" + filenameA + "," + filenameB + "</output>"
print outputStr
```

This will produce an output string that looks something like this "outputFile.jpeg,outputFile.csv". When wrapping this as a service, if you inform the service builder that there are two outputs, these will be parsed into two seperate service outputs.

If the output string contains numeric values, then these must be converted to strings to avoid syntax errors, i.e.

```
    outputStr = "<output>" + filenameA + "," + filenameB + str(F) + "," +
str(F0) + "</output>"
    print outputStr
```

This gives the following (four) output string - "outputFile.jpeg,outputFile.csv,3,0.4"