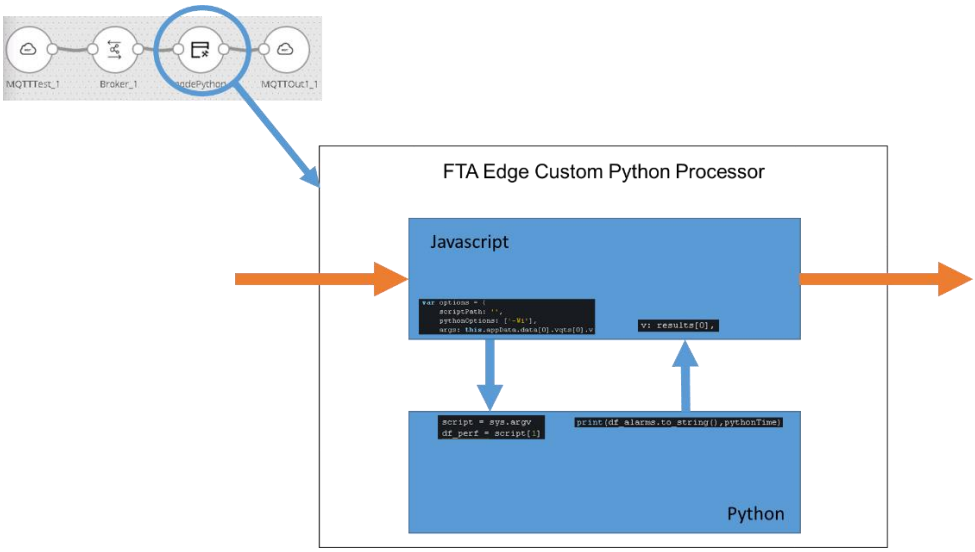
Using Python in Edge

Summary

Custom python scripts may be incorporated into FTAnalytics Edge pipes. With this feature, Edge pipes can provide diverse functionality including complex transformations and orchestration, as well as execution of machine learning algorithms. This document shares the procedure in incorporating Python into a pipe.

Architecture of the custom Python Processor

The figure below illustrates the architecture of the Python processor. The Python processor consists of two parts – a Javascript portion and a Python portion. The Javascript portion handles the passing of data from upstream and downstream processors and allows for that data to be transferred to the Python code. Data payloads entering the Python processor are in JSON format. Python is called from the Javascript code using python-shell. Data may be passed to the Python code by using arguments when calling python-shell. Data from the Python code is passed back to the Javascript code using standard io (e.g. print). More detail for this is provided in the Javascript Template and Javascript/Python Interface section of this document.



Python Environment

For the Python code, any dependent libraries must be installed and available in the run time node. It is a good practice to test the code in the environment to assure that no errors occur prior to integrating to an Edge pipe. As the method of invoking the Python code takes over the standard I/O of Python, warnings and error messages originating from the Python code will create errors in the Edge pipe. To account for this, python-shell has an option to ignore warnings:

```
pythonOptions: ['-Wi']
```

Further details on this is included in the following section.

Javascript Template and Javascript/Python interface

Example Javascript file for Python Processor

The following code is an example of the javascript portion of the custom Python processor. Comments have been highlighted in green below which provide additional information.

```
"use strict";
Object.defineProperty(exports, "__esModule", {value: true});
const cmd = require("node-cmd");
var PythonShell = require('python-shell');
var path = require('path');
const cgp_core_1 = cgp_require("cgp-core");
//-----------------------------------------------------------------------------------------------
//
class nodePython { //the name of the class needs to match the name of the js file
    constructor() {
        this.allInOne_Responses = [];
        this.options = { //options listed here must be created in the custom processor
            fileName: '',
            outputTag: 'NewTag',
            tags: [],
            passAll: true
        }
        this.sendData = [];
        this.appData = {};
        this.historicalData = new Map();
        this.vqtsOut = new Array();
    }
    execute(appData, next) {
        //this function is what is executed by the Edge pipe.  appData is the payload
        //from the previous processor.  next is what is passed to the next processor
        //in the Edge pipe.

        this.appData = appData;
                let that = this;
                var options = {
                        scriptPath: '',
                        args: this.appData.data[0].vqts[0].v //data to be passed to Python
                };
                // args in options are used to pass data into the Python code.
                let shellpath =__dirname.substring(__dirname.indexOf("ShellApp")+9, __dirname.length);
                let filepath = path.join(shellpath, this.options.fileName);
                PythonShell.run(filepath , options, function (err, results) {
                        //PythonShell invokes Python and then runs the custom script
                        // results is what is returned from Python
                        let newTag =  { //create the JSON payload to be passed forward
                                        "@type": "reaction",
                                        id: 'PythonResults',
                                        vqts: [{
                                                v: results[0],
                                                q: 192,
                                                t:(new Date()).toJSON()
                                        }],
                                        reactionType: cgp_core_1.Ia.ReactionType.Value,
                                        mimeType: "",
                                        trackingId: "",
                                        reactionOptions: {},
                                        metadata: {
                                                "@type": "metadata",
                                                "@id": "",
                                                updateRateMs: 0,
                                                updateType: cgp_core_1.Ia.UpdateType.Polling,
                                                reactionOptions: {}
                                        },
                                        links: [{
                                                "@id": ""
```

```
                                        }],
                                        contextPrefix: "",
                                        contextUri: ""
                                    }

                            that.appData.data =  []
                            that.appData.data.push(newTag);
                            next(null, that.appData)
        });
     }
}
exports.nodePython = nodePython; //as with class, must match the name of the js file
```

Example Python script for Python Processor

The code below is sample Python code. Comments have been highlighted in green below which provide additional information.

```
import os
import sys
import psycopg2
import pandas as pd
import numpy as np
import time as tm
import logging
from scipy.stats.stats import mannwhitneyu
from sklearn.utils import resample

# libraries installed in the runtime environment are available

def test_read_from_DB_postgres():  # procedures may be created
    pg_df_perf_Query = "select * FROM performance Where " \
                        "TIME BETWEEN " \
                        "'2019-03-01 00:00:00' AND '2019-03-07 10:00:00'  "
    conn=None
    try:
        conn = psycopg2.connect(host="192.168.1.218", database="postgres",user="postgres",
password="Rockwell2019!")

        # create a cursor
        cur = conn.cursor()

        # execute statements
        df_perf = pd.read_sql_query(pg_df_perf_Query,conn)
        print(df_perf)  # output back to the Edge pipe is sent via standard I/O

    except (Exception, psycopg2.DatabaseError) as error:
        dummy = error

    finally:
        if conn is not None:
            cur.close()
            conn.close()

# input from Edge pipe is brought in via arguments
script = sys.argv
inputFromPipe = script[1]

test_read_from_DB_postgres()
```

Data Exchange between Javascript and Python

As described in the architecture diagram, the data payload flowing through the pipe is handled by the Javascript portion of the custom processor. Data is passed into and out of the python portion of the processor using the following methods:

- Data from Javascript to Python: args in the options (see comment in the code above)
- Data from Python to Javascript: print

Comments in the code above illustrate where the transfer of the data takes place between the Javascript code and Python code.