Python x MATLAB Integration

XGI and Hypernetx Python Toolboxes

Setting Up Virtual Environment

- Utilize the venv command that is in the Python standard library
- Make a folder to contain virtual environment
- From the command line, you run:

  python –m venv C:\path\to\environment\folder

- MATLAB currently only supports up to Python 3.10, so you need to replace '**python**' with the path to the interpreter you need if '**python**' currently runs 3.11 OR you can run the command as:

  py -3.X –m venv C:\path\to\environment\folder

- The X in **-3.X** should be replaced with the desired Python version

Virtual Environment Libraries

- Each virtual environment will contain its own interpreter and installed libraries, so you need to pip install necessary libraries
- First, from the command line, activate the virtual environment:

  <path_to_venv>\Scripts\activate.bat
- Now you can just pip install in the virtual environment
- For hypernetx, must install these two libraries:

  pip install hypernetx

  pip install igraph
- For xgi

  pip install xgi

MATLAB Connection to Virtual Environment

- In MATLAB, running the command '**pyenv**' shows the version of Python MATLAB calls as well as which executable it uses. We want to change this executable to our virtual environment
- In MATLAB, run the command:

  pyenv(Version="path_to_virtual_environment")

- The "**path_to_virtual_environment**" is the path to the Python executable in the virtual environment

Running Python Inside MATLAB

- Can run Python commands in MATLAB using py.

  Example: **variable = py.dict(a = 15, b = 12, c = 8);**

- Can import Python libraries:

  library = py.importlib.import_module('library_name');

- You can then run commands in that library by using library.function_name()

Extra Note for Setup

- It is possible that you need to copy certain folders to a different location to run Python in MATLAB. To check, just try running some Python command and see if you are prompted with errors
- The errors may tell you that you need tcl8.6 and tk8.6 folders in specific file locations. These folders already exist if you have Python installed, you simply need to copy them to the requested locations.
- These folders can be found where Python is installed in your C drive

  AppData\Local\Programs\Python\Python310\tcl\tk8.6

  AppData\Local\Programs\Python\Python310\tcl\tcl8.6

Running Python Files and Integrating with MATLAB

- You can run Python files from MATLAB using pyrunfile()

  variable = pyrunfile('file_name', 'output', input=something)

- '**file_name**' is the Python file name/path to run.
- '**output**' is the name of a variable in the Python script that you want assigned to variable in MATLAB.
- '**input**' is the name of the Python variable you would like to assign the MATLAB variable '**something**' to
- This is the method of sending data back and forth between MATLAB and Python

Handling MATLAB Variables in Python

- The following link is a table of MATLAB to Python data type conversions

https://www.mathworks.com/help/compiler_sdk/python/pass-data-to-matlab-from-python.html

- For instance, if you pass a variable of type double in MATLAB to Python, Python will interpret it as type float
- Passing numeric arrays from MATLAB to Python converts the array to type memoryview. This variable needs its data type updated

Conversion of Memoryview to Usable Arrays

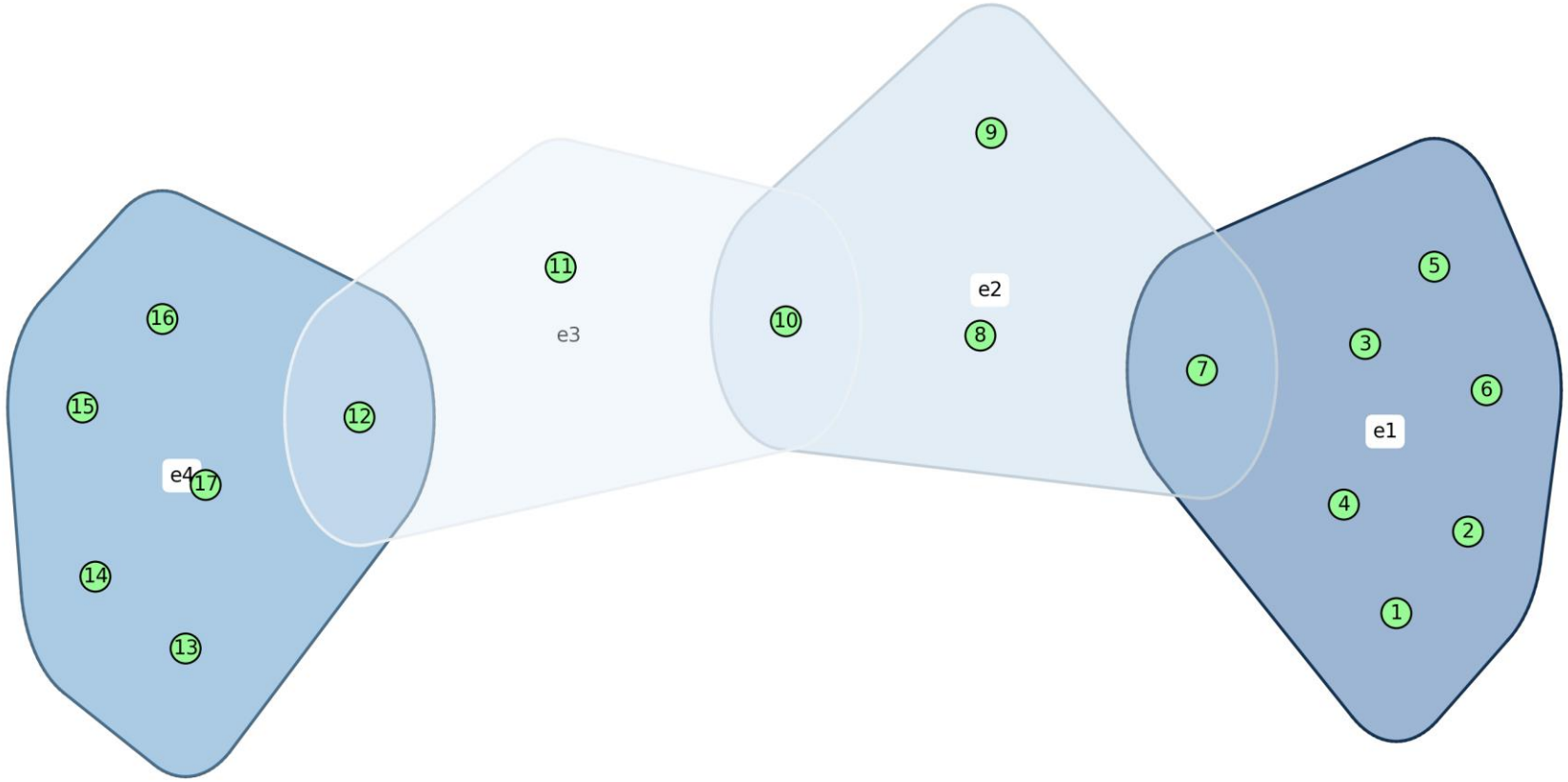
- To change the memoryview variable to something usable, you need to have the '**matlab**' standard library imported

  import matlab

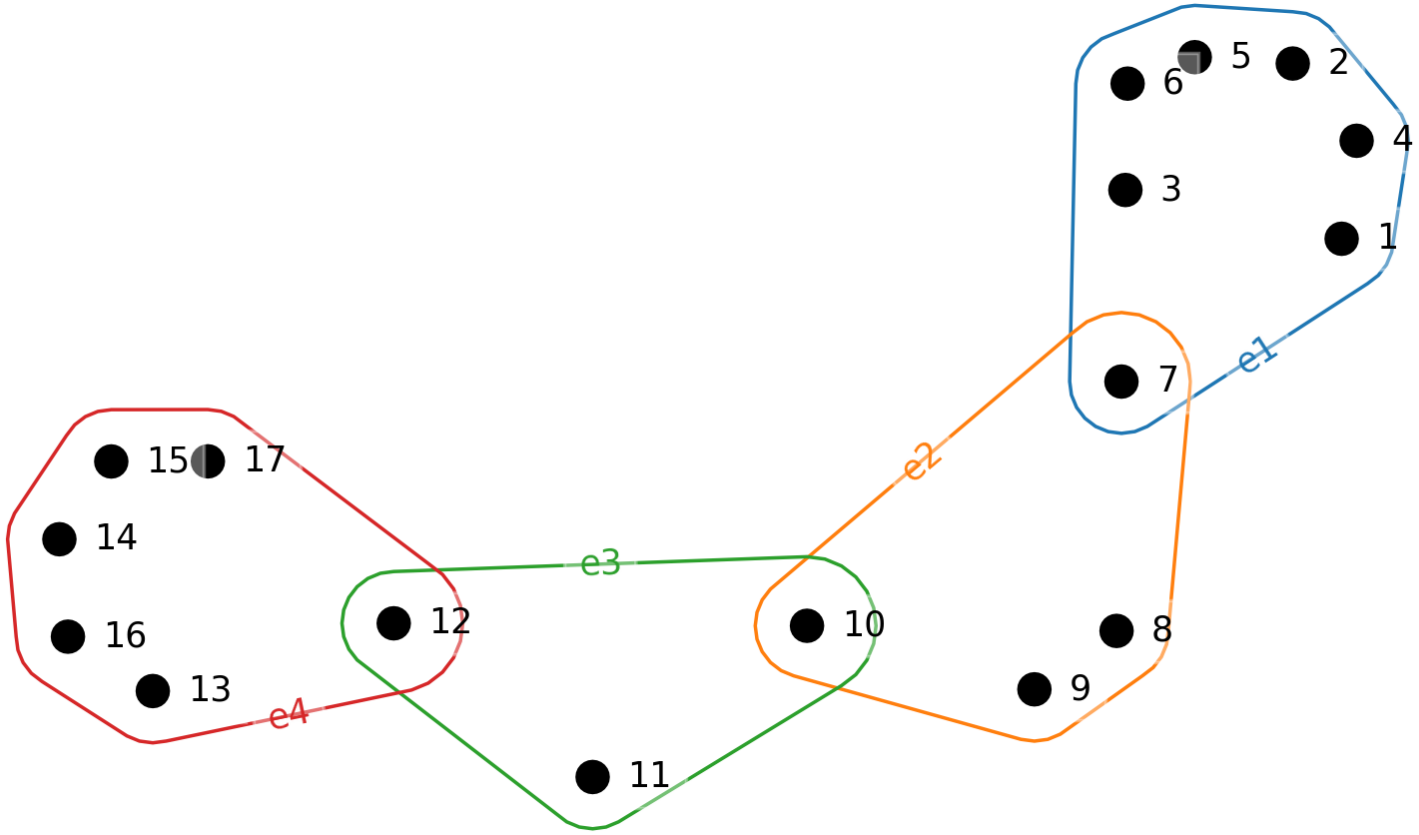
- Then, depending on what data type the array in MATLAB was, you can convert using matlab.data_type(variable)
- For example, if you passed a MATLAB array of type int16 to Python, then you could convert from memoryview using:

  output = matlab.int16(array)

XGI Visualization



Hypernetx Visualization



XGI Shortest Path

- Hypernetx does not possess a shortest path algorithm, but xgi does

```
shortest_paths = pyrunfile('C:\PythonProjects\xgi_test.py', 'shortest_paths', b=B, source_node=1);
```

- **B** is the MATLAB incidence matrix, **b** is the Python variable for that array. **source_node** is a Python variable
- We are returning a Python dictionary with the shortest paths to each other node from node 1

  {1: 0, 2: 1, 3: 1, 4: 1, 5: 1, 6: 1, 7: 1, 8: 2, 9: 2, 10: 2, 11: 3, 12: 3, 13: 4, 14: 4, 15: 4, 16: 4, 17: 4}

- Above is the **shortest_paths** output returned to MATLAB in the form of node: shortest_distance

Running XGI Visualisation

- The xgi_test.py file can be used with MATLAB. If you download the file and use it in **pyrunfile** in MATLAB, you can pass any MATLAB numeric incidence matrix of type int16 and generate a hypergraph visualisation.
- There are several customizations that can be made to the hypergraph including node color, edge color, edge radius, etc. All customizations can be found in the xgi documentation

https://xgi.readthedocs.io/en/stable/index.html