**Session 4 Exercises**

*The first two tasks do not require you to write any new code. They are there to ensure you can assemble the sketches provided in the lecture notes along with the data they parse. This is in preparation for the data challenge (task 4) in which you will extend the sketch to show the data more effectively. Task 3 is optional, but useful for those who wish to practice reading JSON or XML files.*

1. Make sure you can get a parsing sketch working with multiple tabs by copying the simple sketch below from the lecture notes into it. The first block of code should go into your main sketch and the second block (the readData() method) should go into a new tab called parseData. To create a new tab, just click the right-arrow towards the top-right of the Processing window. Don't forget also to drag the file [gdp.tsv](http://staff.city.ac.uk/~jwo/datavis/session04/gdpSketch/data/gdp.tsv) into your sketch so it is available to loadStrings().

|  |
| --- |
| // Reads and displays GDP data.  // Jo Wood, 12th February 2016    float[] years;   // Stores a collection of dates.  float[] gdps;    // Stores a collection of GDP values.    void setup()  {    size(400, 300);    readData();  }    void draw()  {    background(255);  } |

|  |
| --- |
| // Reads the GDP data from a text file.  void readData()  {    String[] textLines =loadStrings("gdp.tsv");      for (String  textLine: textLines)    {      println(textLine);    }  } |

1. Check the sketch works as expected (you should get output similar to the screenshot in the lecture notes showing the contents of the gdp file displayed in the black console at the bottom of your Processing window.
2. Amend the sketch created in Task 1 to display the yearly change in inflation-adjusted GDP per person. Again this requires no new coding as you just copy the amended code from the lecture notes (shown below) into the two tabs of your sketch replacing the code that was previously there.   
   This code goes in your main (left) tab:

|  |
| --- |
| // Reads and displays GDP data.  // Jo Wood, 12th February 2016    float[] years;   // Stores a collection of dates.  float[] gdps;    // Stores a collection of GDP values.    float minYear, maxYear;  float minGDP  =MAX\_FLOAT;  float maxGDP  =MIN\_FLOAT;    void setup()  {    size(400, 200);    readData();    findMinMax();  }    void draw()  {    background(255);      beginShape();   // Start a line shape.    for (int i=0; i<years.length; i++)    {      if (gdps[i] > 0)      {        // Plot each coordinate pair in the line.        float x =map(years[i], minYear, maxYear, 0,width);        float y =map(gdps[i], minGDP, maxGDP,height, 0);        vertex(x, y);      }    }    endShape();   // End the line shape.      noLoop();     // Draw just once.  }    // Finds the minimum and maximum values in the data.  void findMinMax()  {    for (float gdp : gdps)    {      minGDP =min(minGDP, gdp);      maxGDP =max(maxGDP, gdp);    }    // Years are in order, so min/max are first and last elements.    minYear = years[0];    maxYear = years[years.length-1];  } |

1. This code goes in your 'parseData' tab:

|  |
| --- |
| // Reads the GDP data from a text file.  void readData()  {    String[] textLines =loadStrings("gdp.tsv");      // Count the number of data rows in the file.    int numRows = 0;    for (String textLine : textLines)    {      if (!textLine.startsWith("#"))      {        numRows++;      }    }    years =new float[numRows];    gdps  =new float[numRows];      // Read the data into the arrays.    int dataRow = 0;    for (String textLine : textLines)    {      if (!textLine.startsWith("#"))      {        String[] tokens =split(textLine,TAB);          // Read the year data from the first 4 characters in the first column        float date =float(tokens[0].substring(0, 4));          // Read quarter figure from end of the first column and add it to year.        date = date +float(tokens[0].substring(6))/4 - 0.25;          // Store date in the array.        years[dataRow] = date;          // Read the inflation-adjusted GDP per person value from the 6th column.        String gdpText = tokens[5].replace(",","");       // Remove the commas.          if (gdpText.length() == 0)                 // Replace blank cells with 0s.        {          gdpText ="0";        }        gdps[dataRow] =float(gdpText);     // Convert text to a number and store.          // Update the current data row.        dataRow++;      }    }  } |

1. Check it works as expected (displays a simple line graph of the GDP per person over time).
2. *Optional task for more confident coders:* Write a sketch that reads in this simple JSON file [primaries.json](http://gicentre.org/datavis/session04/primariesGenerator/data/primaries.json) and displays some or all of the data graphically (e.g. as rectangles in proportion to the number of delegates coloured by party).
3. [
4. {
5. "winner": "Clinton",
6. "delegates": 23,
7. "state": "Iowa",
8. "party": "Democrat"
9. },
10. {
11. "winner": "Cruz",
12. "delegates": 8,
13. "state": "Iowa",
14. "party": "Republican"
15. },
16. {
17. "winner": "Sanders",
18. "delegates": 15,
19. "state": "New Hampshire",
20. "party": "Democrat"
21. },
22. {
23. "winner": "Trump",
24. "delegates": 10,
25. "state": "New Hampshire",
26. "party": "Republican"
27. }

]

***Hint:*** This is a JSON array containing 4 objects (representing Clinton, Cruz, Sanders and Trump). To read in the array you will need to use [loadJSONArray()](https://processing.org/reference/loadJSONArray_.html) to read the file into a variable of type JSONArray. Each array item will then need to be extracted using [getJSONObject()](https://processing.org/reference/JSONArray_getJSONObject_.html), from which the String and int values can be extracted.   
  
If you have success getting this to work, see if you can do the same, but this time reading the data in XML format from [primaries.xml](http://gicentre.org/datavis/session04/primariesGenerator/data/primaries.xml). To do this you will probably need to refer to the [Processing XML documentation](https://processing.org/reference/XML.html).

1. ***Data Challenge:*** Make a copy of the GDP sketch created in steps 1 and 2 and amend it to provide an improved visualization of GDP over time that attempts to answer the question, *How does the severity and duration of the last economic downturn compare with previous recessions?*. You should consider whether it would be useful to read any of the other data columns found in [gdp.tsv](http://staff.city.ac.uk/~jwo/datavis/session04/gdpSketch/data/gdp.tsv),   
     
   You may also wish to consider whether the use of colour, text labelling and other forms of visual representation might improve your visualization.   
     
   Post the results of your sketch, or links to the sketch code on the discussion board.

**Check on Learning Outcomes**

To ensure that you have achieved the outcomes associated with this session, consider the following, which allow you to evaluate your progress.

 Can I create new tabs in Processing and arrange visualization tasks between them?  
 Can I create and use *arrays* in my Processing sketches?  
 Can I define and call my own *methods* in my Processing sketches?  
 Can I read lines of text from a file into my sketch?  
 Can I split text into tokens to parse data?  
 Do I know where to find documentation in Processing for reading and manipulating in JSON files?  
 Could I read in JSON or XML formatted data into my sketch?  
 Do I know where to look to find data libraries for use with Processing?