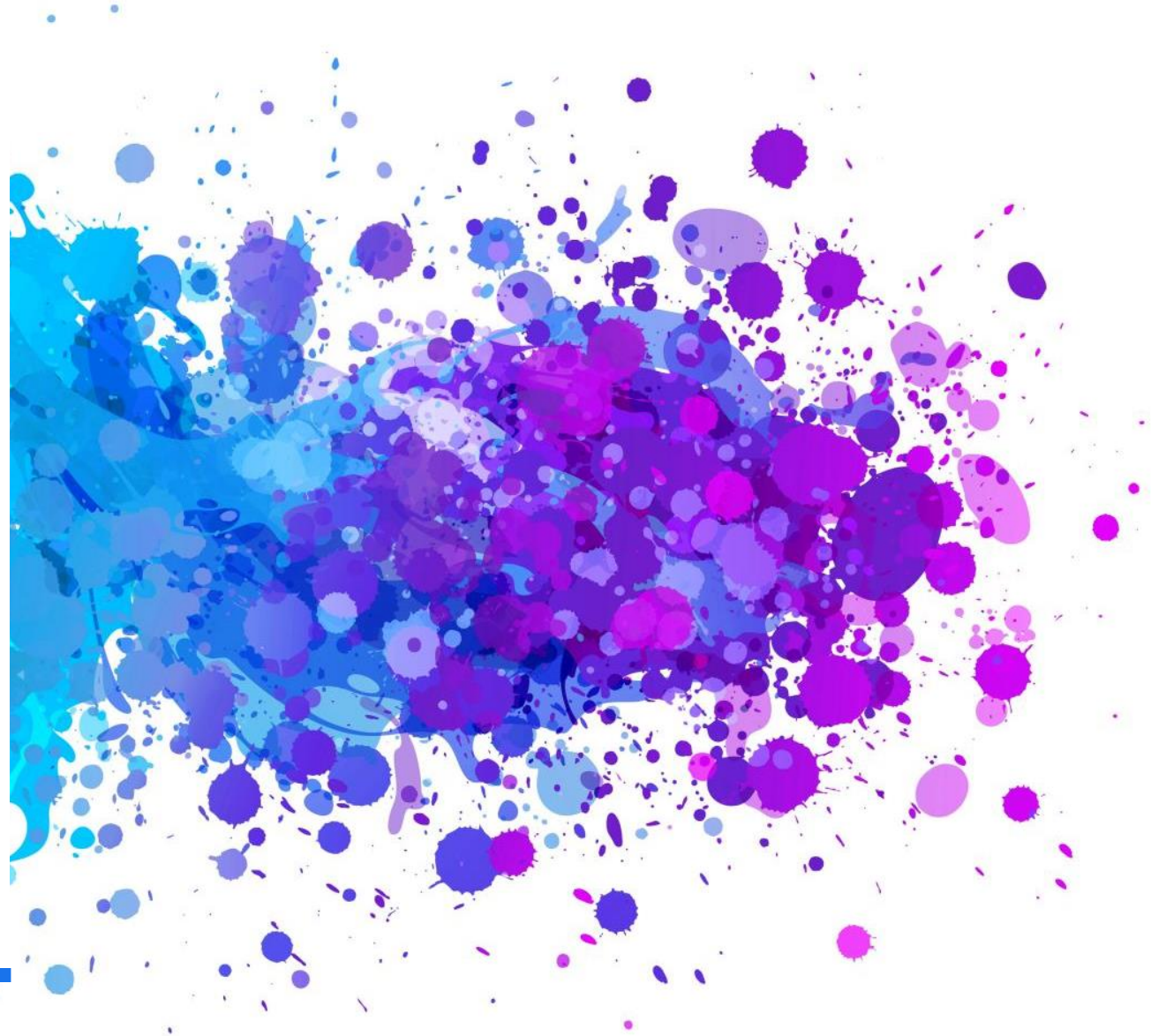
XSLT Batch Processing

WORKING WITH A CORPUS OF XML DOCUMENTS

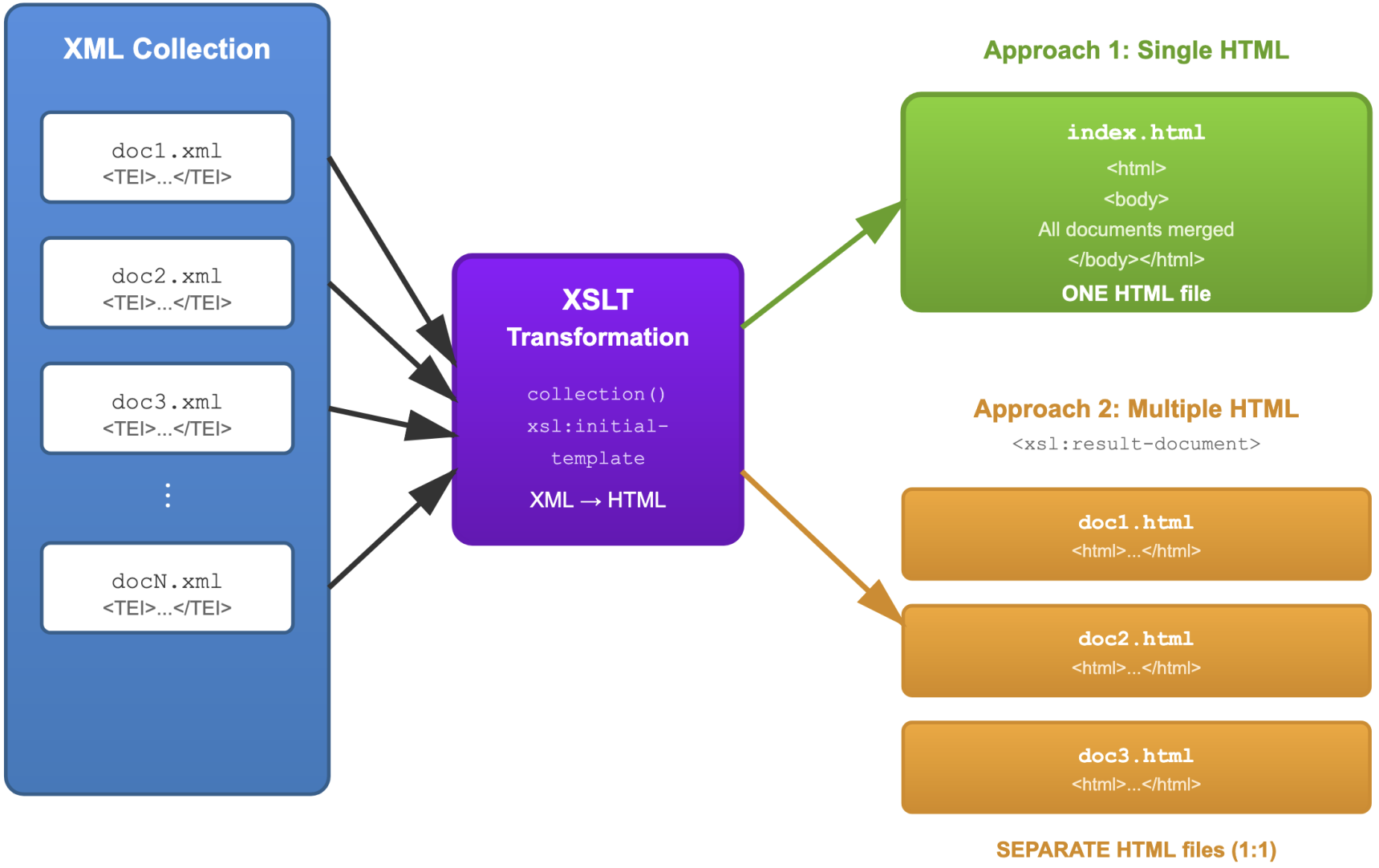


How to...

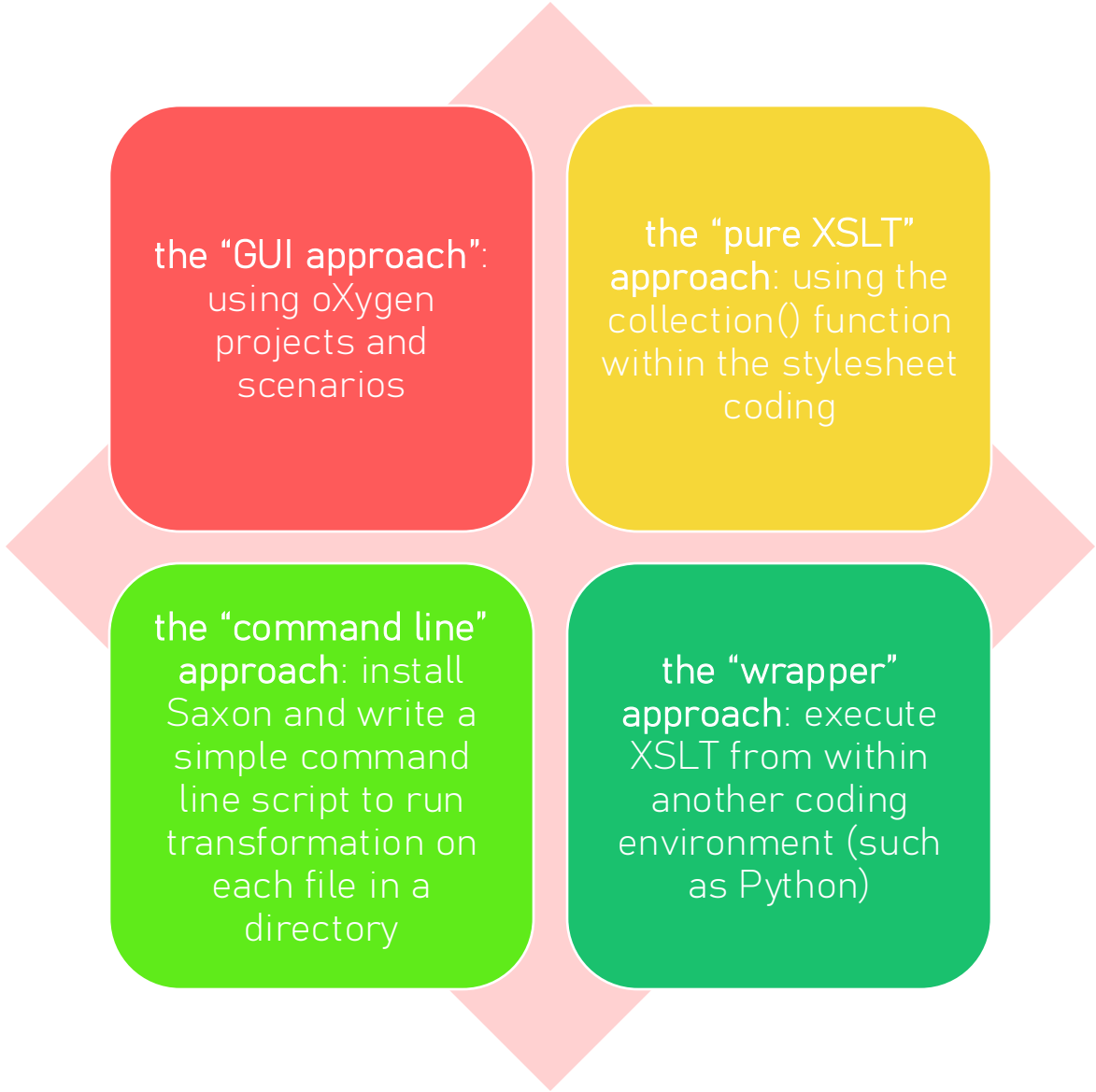
transform each individual XML file in a directory using the same XSLT stylesheet

treat a directory of XML files as a single XML tree to be processed by a single XSLT stylesheet

Two kinds of corpus processing



Approaches:


the “GUI approach”: using oXygen projects and scenarios
the “pure XSLT” approach: using the collection() function within the stylesheet coding
the “command line” approach: install Saxon and write a simple command line script to run transformation on each file in a directory
the “wrapper” approach: execute XSLT from within another coding environment (such as Python)

batch processing within oXygen (the GUI approach)

PROJECTS

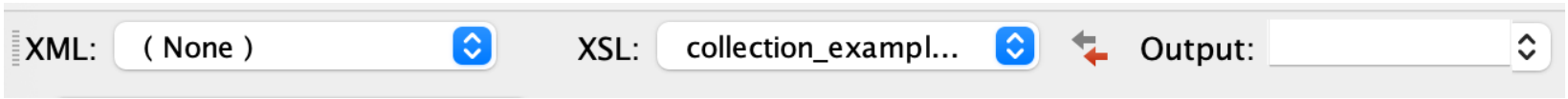
- *What*: A folder system inside oXygen that groups your XML files together
- *Where*: Project menu → New Project (creates a .xpr file)
- *How*: Right-click your project → Add files/folders to populate it
- *Why*: Lets you run the same XSLT transformation on dozens of XML files at once instead of one at a time

SCENARIOS

- *What*: A saved set of instructions that tells oXygen which XSLT to use and where to save the output
- *Where*: Document menu → Transformation → Configure Transformation Scenario
- *How*: Specify your XSLT file, set output location, click OK to save it
- *Why*: Avoids manually setting up the transformation every time you want to process files

batch processing using only XSLT code

- For your homework, you have been using the oXygen debugger to specify *path* locations of your input XML document, your XSLT stylesheet, and your output (often HTML) file using dropdown menus.



- But you could specify all of that information from within the XSLT code itself. For instance:

```
<xsl:variable name="single-doc" as="document-node()"
    select="doc('./path/to/input.xml')"/>

<xsl:template name="xsl:initial-template">
    <root>
        <xsl:apply-templates select="$single-doc"/>
    </root>
</xsl:template>
```

batch processing using only XSLT code

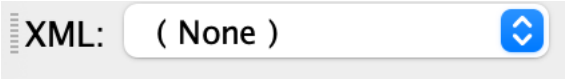
- declare variable with path to directory within which your XML documents are kept
- <xsl:template name="xsl:initial-template">: this overrides the default input document

because there is no default input document, you need to specify that you are pointing at your variable with the initial <xsl:apply-templates select="$mitford-corpus"/>

```
<xsl:variable name="mitford-corpus" as="document-node()+"
    select="collection('./mitford_test_collection?recurse=yes;select=*.xml')"/>
<xsl:template name="xsl:initial-template">
    <root>
        <metadata>There are <xsl:value-of select="$mitford-corpus//placeName =>
            count()"/> places in the
            corpus.</metadata>
        <xsl:apply-templates select="$mitford-corpus"/>
    </root>
</xsl:template>
<xsl:template match="/">
    <d>Found a document!</d>
</xsl:template>
```

additional points / some notes of caution

- the *collection*(‘directory_path’) is finicky, so proceed deliberately and carefully:
  - if there are *any* files in the directory that are not well-formed XML, you will get a cryptic error message
- for your html superstructure, you can no longer match by the root element, because now you have many different root elements in the collection. hence <xsl:template name="xsl:initial-template">
- when you apply templates, you can no longer assume an input document the way you did in homework: you have to be more specific. hence <xsl:apply-templates select="$mitford-corpus"/>
- the XML input must be set to none, otherwise that setting will override the inline code



applying the same XSLT stylesheet to each XML file within a directory

- with <xsl:result-document>, you can specify where you want to save an output file
- which allows you to batch process directories of input files

```
<xsl:for-each select="$mitford-corpus">
    <xsl:result-document href="output/file_{position()}.xml">
        <xsl:apply-templates select="."/>
    </xsl:result-document>
</xsl:for-each>
```

run XSLT transformations (batch or otherwise) from command line

1. write XSLT stylesheet in an editor (oXygen or VS Code), as you normally would
2. download Saxon HE (the same engine oXygen uses to run XSLT) Java file to your hard drive: https://www.saxonica.com/html/download/java.html
3. command line syntax to run the transformation:

```
java -cp "path_to_local_saxon_java" net.sf.saxon.Transform -s:"path_to_XML" -
    xsl:"path_to_XSLT_stylesheet" -o:"path_to_output"
```

applying command line java transformation to multiple files

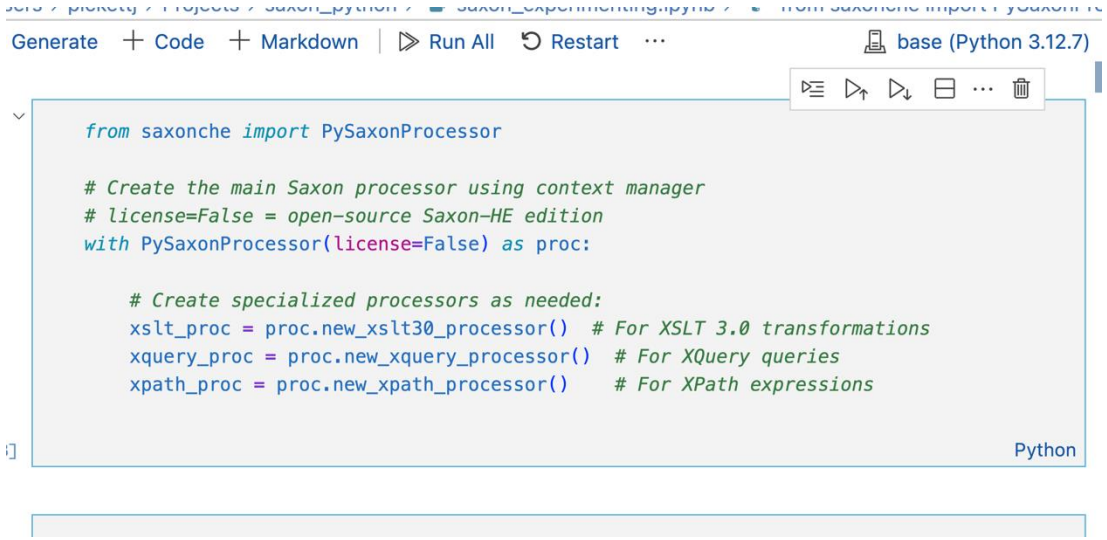
```
for file in input_directory/*.xml; do
    filename=$(basename "$file")
    outputname="${filename%.xml}.html"

    java -jar saxon-he-12.5.jar \
        -s:"$file" \
        -xsl:"path_to_XSLT_stylesheet" \
        -o:"output_directory/$outputname"
done
```

- simple “for loop,” same logic as in XSLT: “for each of these items, do something”
- *filename* and *outputname* are temporary variables that change for each file in the input directory, which we use to save separately titled outputs
  - *basename* is a Linux function that strips away the full path, keeps only the file name

fourth alternative: running XSLT from within another coding environment (such as Python)

- Python offers a Saxon “wrapper” library that allows implementation of XSLT (and XPath, and XQuery).
- This allows for batch processing using standard Python language: for each item, run this transformation.
- Not much advantage to this method *unless* you are integrating your XML with other Python capabilities.



you shouldn't be using the oXygen debugger to individually transform each XML file in a directory

BOTTOM LINE