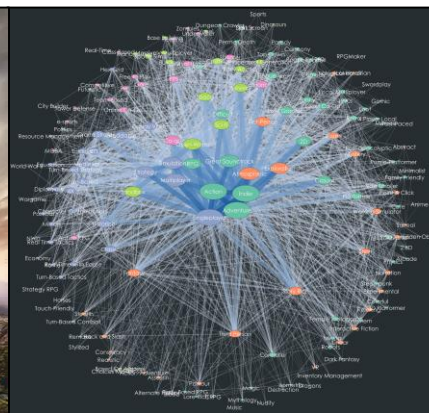




Universiteit
Leiden

Digital Approaches to Historical Inquiries

5th Class



Assignment III

- Remember: these federal employees are not even supposed to accept gifts from foreign sources!
- End goal of this assignment is to have an exhaustive list of all the reasons these people have for still accepting gifts.
- Aim is to be able to extract information from an XML file as well as find solution to a programming question online.
- Spend 2 hours on this
- By next Tuesday, hand in, via a direct message to me in Slack:
 - your solution/the current output of your script (even if it is an error)
 - your script (even if it does not function)
 - a short overview of how you got to where you did (resources used, alternative solutions tried, etc.)

How many reasons?

How do I love thee?
Let me count the ways.
One, one thousand.
Two, one thousand.
Three, one thousand.



Obscure “Who Framed Roger Rabbit” reference. Little to do with the actual assignment, but what can I say... I just love that movie!

I WANT TO PLAY A



**SUPERCALIFRAGILISTICEXPIALIDOCIOUS
GAME**

Programming is faster

- How many letters in: **supercalifragilisticexpialidocious**
- In case I lose, I mean: generally speaking! ;-)

Assignment III, another, faster solution (also... XPath)

```
import xml.etree.ElementTree as etree
with open('ObamaGifts.xml', 'rb') as xml_file:
    tree = etree.parse(xml_file)

root = tree.getroot()
reasons = root.findall("./REASON")


for reason in reasons:
    if "embarrassment" in reason.text: continue
    else:
        print ("not embarrassment: " + reason.text)
```

Select current node (root)

↑
./REASON

Selects nodes from the current node
that match the selection.

(No matter where they are.)



"A gift? For me?
You shouldn't have...
Really, I mean it!"

Assignment III, another solution that gets you the computationally correct answer.

```
import xml.etree.ElementTree as etree
with open('ObamaGifts.xml', 'rb') as xml_file:
    tree = etree.parse(xml_file)

root = tree.getroot()
reasonlist = list() #create a
reasons =root.findall("./REASON")#use findall() and XPath to select
all (//) elements that are under the current node (. , in this case
the root) that have the GIFT tag and put them in a list

for reason in reasons: #for every gift element that is in the gifts
list, do the following
    nitpicking = reason.text
    if nitpicking in reasonlist: continue
    reasonlist.append(nitpicking)
print (reasonlist)
```



When it comes to coding,
the devil is in the details



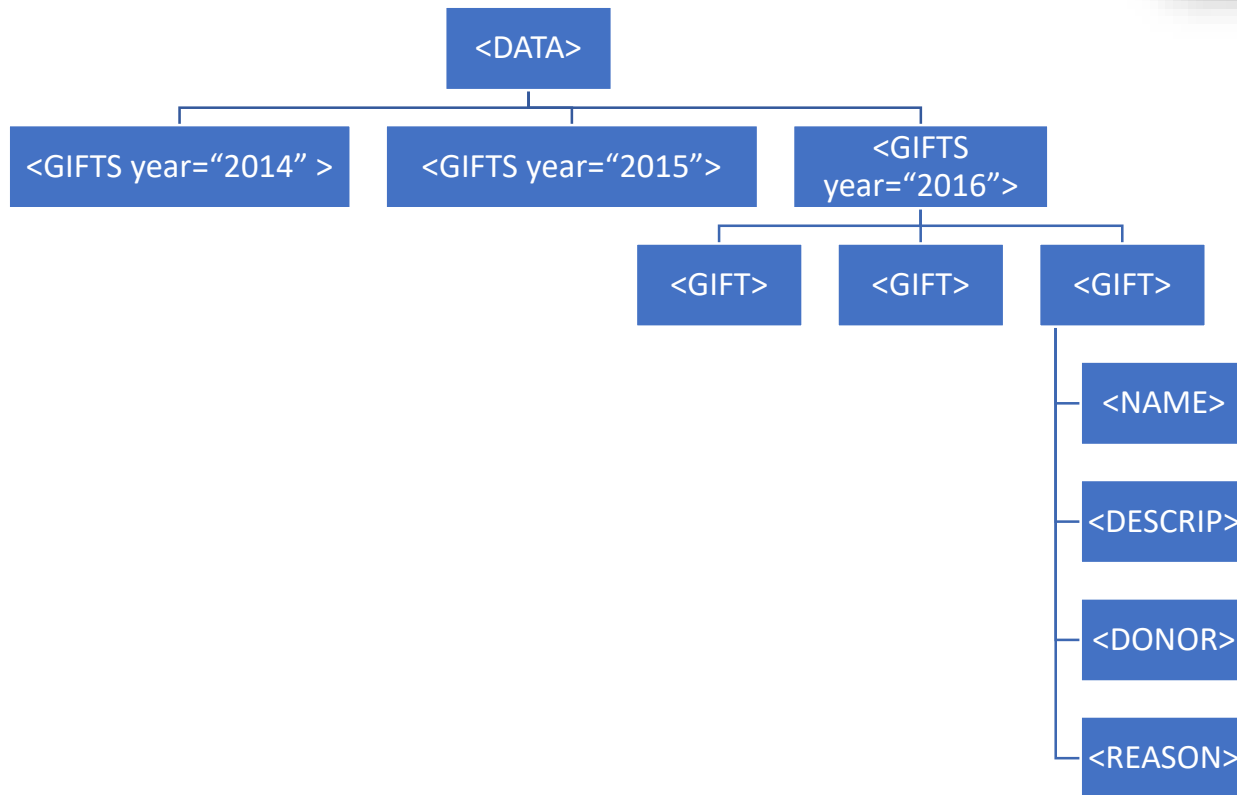
- Anthropology of the Gift (post-Maussian)
 - *Essai sur le don* (Marcel Mauss 1924/1925)
 - Why do we give presents, accept them, and return them?
- Social and Material
 - Ties that Matter, Matter that Ties
- About us as a species!
- About self, about others!
- About communication!
- About cooperation
- About competition!
- About conspicuous consumption!
- About individuals, communities, or states!



Marcel Mauss



Back to Obama's Gift Tree!



Let's unwrap that gift description!

<GIFT>

<NAME>The Honorable Barack Obama, President of the United States</NAME>

<DESCRIP>6" cast bronze medal depicting vines on ceramic stoneware with a Roman travertine finish. 8" cast bronze medal depicting the Angel of Solidarity and Peace. Book, title: Miserando Atque Eligando. Rec'd-3/27/2014. Est. Value-\$1,015.00. Disposition-National Archives and Records Administration</DESCRIP>

<DONOR>His Holiness Pope Francis, Holy See</DONOR>

<REASON>Non-acceptance would cause embarrassment to donor and U.S. Government.</REASON>

</GIFT>

- Describes one “gift event”
- Has extended object descriptions
- Can hold multiple objects
- Has a date stamp
- Has a monetary value
- Has a place of disposition



Hannah, my daughter, unwrapping a gift.

Obligatory tree joke: The apple doesn't fall from the tree!

Regular Expressions

- Brief refresher:
 - A sequence of characters that define a search term
 - A mathematical notation to formally describe and classify language
 - Basically, a really good, really multi-functional search tool
- If there is one thing you learn from this course/minor, it should be regular expressions.
 - <https://regexone.com/>
- Downsides:
 - Can be fiddly
 - Can feel like they give the right answer
 - “To someone with a hammer, everything looks like a nail”
 - Are computationally slower than other methods



Image source: [xkcd](https://xkcd.com/)

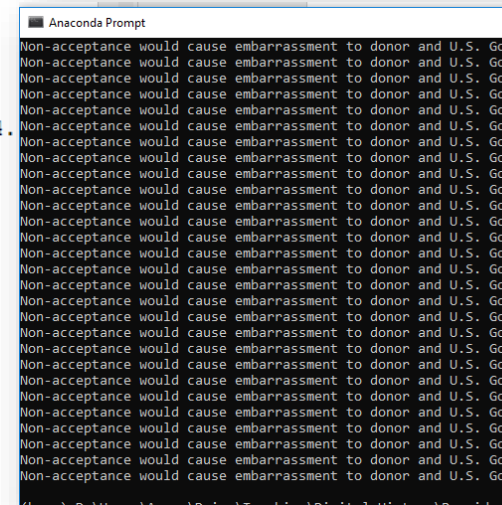


Assignment 3, a regex variant of the popular vote

```
import re

with open('ObamaGifts.xml', 'r', encoding="utf8") as file:
    text = file.read()
reasons = re.findall('<REASON>(.*?)</REASON>', text)
for reason in reasons:
    print (reason)
```

```
<GIFT>
<NAME>The Honorable Barack Obama, President of the United States</NAME>
<DESCRIP>6" cast bronze medal depicting vines on ceramic stoneware with a Roman travertine finish. 8" cast
bronze medal depicting the Angel of Solidarity and Peace. Book, title: Miserando Atque Eligando. Rec'd-3/27/2014.
Est. Value-$1,015.00. Disposition-National Archives and Records Administration</DESCRIP>
<DONOR>His Holiness Pope Francis, Holy See</DONOR>
<REASON>Non-acceptance would cause embarrassment to donor and U.S. Government.</REASON>
</GIFT>
```





Step 0, redux!

- Download this weeks how-to zip folder from the syllabus' Handling Historical Data 2
 - Unzip it in your work directory
- Open the anaconda command prompt.
- Navigate to your work directory

```
p: #navigate to your p: drive
cd #access folder
dir # see contents of folder
cd .. # go up one folder
cls # clear console screen
python helloworld.py # run a python program in Anaconda
```

Regex how-to

- Go to www.pythex.org and...
- Try it yourself:
 - extract (match) the dollar amounts from the description.
 - Hint: you can escape any special characters (characters reserved by regex to do stuff with) by putting a `\` in front
- Use this as a test string (you can find it in `descriptest.txt`):

36" × 28" framed facsimile letter to Abraham Lincoln, dated June 4, 1863, from Henry Parks and the people of Sydney, Australia. Billiards cue made of American and Australian wood in carrying case. Rec'd—1/19/2016. Est. Value—\$515.00. Disposition—National Archives and Records Administration

White linen set, hand-knit in the Ao po'i Paraguayan style, including large table cloth, two small table coverings, apron and napkins. Rec'd—1/28/2016. Est. Value—\$560.00. Disposition—National Archives and Records Administration

Three bottles of Italian wine and carrier box, Florence-made, of maple and burgundy leather, with a reproduction of the lithograph The Montecavallo Square by Philippe Benoise. Rec'd—2/8/2016. Est. Value—\$667.00. Disposition—National Archives and Records Administration. Wine handled pursuant to U.S. Secret Service policy

Step 9: Printing money from a tree!

```
import xml.etree.ElementTree as etree
import re
with open('ObamaGifts.xml', 'rb') as xml_file:
    tree = etree.parse(xml_file)

root = tree.getroot()
gifts = root.findall("./GIFT")
for gift in gifts:
    if "Obama" in gift[0].text:
        descrip = gift[1].text
        print (descrip)
        money = re.findall('\$\d{1,3}', descrip)
        print(money)
```

- Try it yourself:
 - run step9.py



Step 10: checking up on regex



```
import xml.etree.ElementTree as etree
import re
with open('ObamaGifts.xml', 'rb') as xml_file:
    tree = etree.parse(xml_file)
root = tree.getroot()
reasonlist = list()
gifts = root.findall("./GIFT")
for gift in gifts:
    if "Obama" in gift[0].text:
        descrip = gift[1].text
        value = re.findall('\$[\d,]+', descrip)
        if len(value) > 1:
            print(descrip)
        if len(value) == 0:
            print(descrip)
print ("Done running script")
```

- Answer this:
 - What regex could you use to make (even more) sure you only get one estimated value per element?
 - Change the last print function to print(value). What does it print? Why?

Step 11: stuffing your wallets



```
#in the step11.py there is more code above
for gift in gifts:
    if "Obama" in gift[0].text:
        descrip = gift[1].text
        money = re.findall('\$[\d,]+', descrip)
        for item in money:
            wallet.append(item)

print (wallet)
```

- Answer this:
 - How can you (computationally) check the type/class of Python object the wallet is?
 - Hint, you can find the type/class of a python object with the type() function.
 - Hint2 don't forget to print the results of the function!
 - What is the value of all the "money" in the wallet?
 - Hint: summing is done with sum()
 - What is the result?

Step 12: making printable wallets out of handbags or... integers out of strings... and back again

```
#in the script there is more code above this
wallet = list()

for gift in gifts: #for every gift element that is in the gifts list, do the following
    if "Obama" in gift[0].text:
        descrip = gift[1].text
        #print (descrip)
        handbag = re.findall('\$[\d,]+', descrip)
        for item in handbag:
            money = item.replace("$", "").replace(",", "")
            print (money)
            try:
                money = int(money)
            except:
                print ("Item is not an integer!")
            wallet.append(money)

try:
    print ("The total value of all gifts to Obama is "+str(sum(wallet)))
except:
    print ("Done, but one or more items were not converted to integers")
```

- Run the script (python step12.py)
- Answer this:
 - What else do we need to .replace to transform every string to an integer?



Step 12: making wallets out of handbags or... integers out of strings.

```
#in the script there is more code above this
wallet = list()

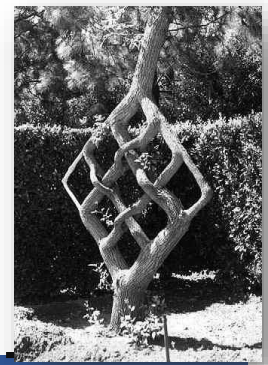
for gift in gifts: #for every gift element that is in the gifts list, do the following
    if "Obama" in gift[0].text:
        describ = gift[1].text
        #print (describ)
        handbag = re.findall('\$[\d,]+', describ)
        for item in handbag:
            item = item.replace("$", "")
            print (item)
            try:
                item = int(item)
            except:
                print ("Item is not an integer!")
            wallet.append(item)

try:
    print ("Total value of all gifts is $" + sum(wallet))
except:
    print ("Done, but one or more items were not converted to integers")
```

- Run the script (python step12.py)
- Answer this:
 - What else do we need to .replace to transform every string to an integer?



Step 13: adding stuff to the tree



```
import xml.etree.ElementTree as etree
import re
with open('ObamaGifts.xml', 'rb') as xml_file:
    tree = etree.parse(xml_file)

root = tree.getroot()
gifts =root.findall("./GIFT")#use findall() and XPath to select all (//) elements that are under the
current node (. , in this case the root) that have the GIFT tag and put them in a list

valuelist = list()

for gift in gifts: #for every gift element that is in the gifts list, do the following
    if "Obama" in gift[0].text:
        descrip = gift[1].text
        handbag = re.findall('\$[\d,]+', descrip)
        for item in handbag:
            money = item.replace("$", "").replace(",","")
            xmlval = etree.SubElement(gift, 'VALUE')
            xmlval.text = money

mydata = etree.tostring(root)
with open('ObamaGifts2.xml', 'wb') as xml_file:
    xml_file.write(mydata)
```

- Run the script
- Answer this:
 - What is the result in the ObamaGifts2.XML file?
 - Hint: there are three things a bit off...

Step 14: Making an Obama tree



```
import lxml.etree as etree
import re
parser = etree.XMLParser(remove_blank_text=True)
with open('ObamaGifts.xml', 'rb') as xml_file:
    tree = etree.parse(xml_file, parser)

root = tree.getroot()
gifts = root.findall("./GIFT")
wallet = list()

for gift in gifts: #for every gift element that is in the gifts list, do the following
    if "Obama" in gift[0].text:
        descrip = gift[1].text
        #print (descrip)
        handbag = re.findall('\$[\d,]+', descrip)
        for item in handbag:
            money = item.replace("$", "").replace(",", "")
            xmlval = etree.SubElement(gift, 'VALUE')
            xmlval.text = money
    else:
        giftparent = gift.getparent()
        giftparent.remove(gift)
mydata = etree.tostring(root, pretty_print=True, encoding='UTF-8')
with open('OnlyObamaGifts.xml', 'wb') as xml_file:
    xml_file.write(mydata)
```

- Run the script
- Answer this:
 - What is the result in the ObamaGifts2.XML file?
- Try it yourself:
 - Add “currency” attribute to <VALUE> and set its value to “U.S. dollars”

FAIR Principles (Wilkinson et al. 2016)

- To be Findable:

FAIR Principles (Wilkinson et al. 2016)

- To be Findable:
 - F1. (meta)data are assigned a globally unique and persistent identifier
 - F2. data are described with rich metadata (defined by RI below)
 - F3. metadata clearly and explicitly include the identifier of the data it describes
 - F4. (meta)data are registered or indexed in a searchable resource
- To be Accessible:

FAIR Principles (Wilkinson et al. 2016)

- To be Findable:
 - F1. (meta)data are assigned a globally unique and persistent identifier
 - F2. data are described with rich metadata (defined by R1 below)
 - F3. metadata clearly and explicitly include the identifier of the data it describes
 - F4. (meta)data are registered or indexed in a searchable resource
- To be Accessible:
 - A1. (meta)data are retrievable by their identifier using a standardized communications protocol
 - A1.1 the protocol is open, free, and universally implementable
 - A1.2 the protocol allows for an authentication and authorization procedure, where necessary
 - A2. metadata are accessible, even when the data are no longer available
- To be Interoperable:

FAIR Principles (Wilkinson et al. 2016)

- To be Findable:
 - F1. (meta)data are assigned a globally unique and persistent identifier
 - F2. data are described with rich metadata (defined by R1 below)
 - F3. metadata clearly and explicitly include the identifier of the data it describes
 - F4. (meta)data are registered or indexed in a searchable resource
- To be Accessible:
 - A1. (meta)data are retrievable by their identifier using a standardized communications protocol
 - A1.1 the protocol is open, free, and universally implementable
 - A1.2 the protocol allows for an authentication and authorization procedure, where necessary
 - A2. metadata are accessible, even when the data are no longer available
- To be Interoperable:
 - I1. (meta)data use a formal, accessible, shared, and broadly applicable language for knowledge representation.
 - I2. (meta)data use vocabularies that follow FAIR principles
 - I3. (meta)data include qualified references to other (meta)data
- To be Reusable:

FAIR Principles (Wilkinson et al. 2016)

- To be Findable:
 - F1. (meta)data are assigned a globally unique and persistent identifier
 - F2. data are described with rich metadata (defined by R1 below)
 - F3. metadata clearly and explicitly include the identifier of the data it describes
 - F4. (meta)data are registered or indexed in a searchable resource
- To be Accessible:
 - A1. (meta)data are retrievable by their identifier using a standardized communications protocol
 - A1.1 the protocol is open, free, and universally implementable
 - A1.2 the protocol allows for an authentication and authorization procedure, where necessary
 - A2. metadata are accessible, even when the data are no longer available
- To be Interoperable:
 - I1. (meta)data use a formal, accessible, shared, and broadly applicable language for knowledge representation.
 - I2. (meta)data use vocabularies that follow FAIR principles
 - I3. (meta)data include qualified references to other (meta)data
- To be Reusable:
 - R1. meta(data) are richly described with a plurality of accurate and relevant attributes
 - R1.1. (meta)data are released with a clear and accessible data usage license
 - R1.2. (meta)data are associated with detailed provenance
 - R1.3. (meta)data meet domain-relevant community standards

What do you think of the need for FAIR data?

- How would it apply in particular to historical inquiries?
- Licensing data or something else? Check:
 - <https://opendatacommons.org/licenses/>
 - <https://creativecommons.org/licenses/>
 - <https://opensource.org/licenses>

Assignment 4

- A “mini-FAIR approach” to showing what you can do with Python and XMLs.
 - Findable for others in this course
 - Accessible for others in this course
 - Interoperable by others in this course
 - Re-usable by others (that, in principle means a license!)
- Spend max. three hours on doing this assignment, and 1 hour on keeping a log/writing a reflection.
- Depending on your (self-assessed) skill level with Python:
 - Do assignment 4a
 - Do assignment 4b
- Graded on mini-FAIR principles and progress relative to current level.

Assignment 4a

- Make sure you are able to run through all the steps we have taken over the last two weeks and understand what the code does.
 - To make it a bit more difficult, I did not add commenting to steps 8-14,
 - Google functions, variables, etc. if you are not sure if you understand them.
 - If you need to focus on learning more python, do so with an online tutorial.
 - Provide a written reflection on what you learned in the last two weeks, how you learned this, what challenges you are still experiencing as well as what lessons others could draw from your Python exploring
 - Provide a working script that showcases your “state of the art” Python programming skills
 - NB you have written this script yourself.
 - Even if it is just “Hello World”
 - Submit all your (meta-)data (reflection and your script as well as any other files) in a zip file in Slack, in the digitalapproaches channel (or mail them to me).

Assignment 4b

- Extract all the dates from the descriptions of gifts to Obama and stick it into an XML (using regex)
- If done, keep extracting other information from the description field (e.g. disposition, individual gifts) and stick them into the XML.
- If done, extract donor countries from the donor element in the geonames gazetteer format (www.geonames.org).
- If done, do the same for the Bush era (from the original files in the)
- If done, do the same for all the other presidents
- If done, write a paper with me on the topic of foreign gifts to American presidents
- If done, continue your research career all the way until you are a professor.
- If done, create world peace through your gift research.

- Keep a log that shows how you did this and what challenges you encountered.
- Submit your (meta-)data, i.e. log, and all your scripts, xmls and all other files, in a zip file in Slack, in the digitalapproaches channel (or mail them to me).

See you next week!

I'll send you a course survey via Google Forms again.

Please take the time to fill it out before the end of next week

Next week:

- No open office (same for the week after)
- Guest: