This document is organized in 3 sections:

- Recap of Eli Bendersky's XML parsing from his online article
- Parsing and helper functions for Kanjidic2
- Parsing and helper functions for JMDict

Eli Bendersky's article tests

Basic parsing

```
<?xml version="1.0"?>
<doc>
    <branch name="testing" hash="1cdf045c">
        text,source
    </branch>
    <branch name="release01" hash="f200013e">
        <sub-branch name="subrelease01">
            xml,sgml
        </sub-branch>
    </branch>
    <branch name="invalid">
    </branch>
</doc>
```

```
In [96]: import xml.etree.cElementTree as ET
         tree = ET.ElementTree(file='doc1.xml')
```

```
In [80]: tree.getroot()

Out[80]: <Element 'doc' at
         0x05349EC0>
```

```
In [83]: root = tree.getroot()
         root.tag, root.attrib

Out[83]: ('doc',
         {})
```

```
In [84]: for child_of_root in root:
             print child_of_root.tag, child_of_root.attrib

        branch {'hash': '1cdf045c', 'name': 'testing'}
        branch {'hash': 'f200013e', 'name': 'release01'}
        branch {'name': 'invalid'}
```

```
In [85]: root[0].tag, root[0].text

Out[85]: ('branch', '\n        text,source\n
         ')
```

Find interesting stuff

```
In [86]: for elem in tree.iter():
             print elem.tag, elem.attrib
```

```
doc {}
branch {'hash': '1cdf045c', 'name': 'testing'}
branch {'hash': 'f200013e', 'name': 'release01'}
sub-branch {'name': 'subrelease01'}
branch {'name': 'invalid'}
```

```
In [87]: for elem in tree.iter(tag='branch'):
             print elem.tag, elem.attrib
```

```
branch {'hash': '1cdf045c', 'name': 'testing'}
branch {'hash': 'f200013e', 'name': 'release01'}
branch {'name': 'invalid'}
```

Using XPath

```
In [97]: for elem in tree.iterfind('branch/sub-branch'):
             print elem.tag, elem.attrib
```

```
sub-branch {'name': 'subrelease01'}
```

```
In [99]: for elem in tree.iterfind('branch'):
             print elem.tag, elem.attrib
```

```
branch {'hash': '1cdf045c', 'name': 'testing'}
branch {'hash': 'f200013e', 'name': 'release01'}
branch {'name': 'invalid'}
```

```
In [90]: for elem in tree.iterfind('branch[@name="release01"]'):
             print elem.tag, elem.attrib
```

```
branch {'hash': 'f200013e', 'name': 'release01'}
```

Kanjidic2

Example file

```
In [105]: import xml.etree.cElementTree as ET
          tree = ET.ElementTree(file='kanjidic2_example.xml')
```

First of all, what does the tree look like in this example file?

```
In [25]: elems = [elem for elem in tree.iter()][:10]
         elems
```

```
Out[25]: [<Element 'character' at
         0x050E6848>,
          <Element 'literal' at 0x050E68A8>,
          <Element 'codepoint' at
         0x050E68D8>,
          <Element 'cp_value' at
         0x050E6908>,
          <Element 'cp_value' at
         0x050E6920>,
          <Element 'radical' at 0x050E6938>,
          <Element 'rad_value' at
         0x050E6968>,
          <Element 'rad_value' at
         0x050E6980>,
          <Element 'misc' at 0x050E6998>,
          <Element 'grade' at 0x050E69C8>]
```

Getting the root: the 'character'.

```
In [15]: root = tree.getroot()
         root
```

```
Out[15]: <Element 'character' at
         0x050E6848>
```

Getting the literal.

```
In [16]: literal = root[0]
         literal
```

```
Out[16]: <Element 'literal' at
         0x050E68A8>
```

```
In [17]: kanji = literal.text
         kanji
```

```
Out[17]: u'\u672c'
```

```
In [18]: print kanji
```

```
本
```

Getting the meanings.

```
In [44]: meanings = [elem for elem in tree.iter('meaning')]
         [meaning.text for meaning in meanings]
```

```
Out[44]: ['book',
          'present',
          'main',
          'true',
          'real',
          'counter for long cylindrical
         things',
          'livre',
          u'pr\xe9sent',
          'essentiel',
          'origine',
          'principal',
          u'r\xe9alit\xe9',
          u'v\xe9rit\xe9',
          u"compteur d'objets allong\xe9s",
          'libro',
          'origen',
          'base',
          'contador de cosas alargadas',
          'livro',
          'presente',
          'real',
          'verdadeiro',
          'principal',
          'sufixo p/ contagem De coisas
         longas']
```

But here we only want english meanings.

```
In [46]: meanings[10].attrib
```

```
Out[46]: {'m_lang':
         'fr'}
```

```
In [51]: english_meanings = filter(lambda elem: elem.attrib == {}, meanings)
         [meaning.text for meaning in english_meanings]
```

```
Out[51]: ['book',
          'present',
          'main',
          'true',
          'real',
          'counter for long cylindrical
         things']
```

Finally, we can get the Kanas.

```
In [60]: readings = [elem for elem in tree.iter('reading')]
         print [reading.text for reading in readings]
```

```
['ben3', 'bon', u'\ubcf8', u'\u30db\u30f3', u'\u3082\u3068']
```

Filtering for kanas.

```
In [64]: readings[0].attrib['r_type']
```

```
Out[64]: 'pinyin'
```

```
In [67]: 'r_type' in readings[0].attrib

Out[67]: True

In [73]: kanas = filter(lambda reading: reading.attrib['r_type'] in ['ja_on', 'ja_kun'], rea
         kanas

Out[73]: [<Element 'reading' at 0x050E6EF0>, <Element 'reading' at
         0x050E6F08>]

In [75]: for kana in kanas:
             print kana.text

         ホン
         もと
```

The whole Kanjidic2 file

```
In [137]: import xml.etree.cElementTree as ET
          tree = ET.ElementTree(file='kanjidic2.xml')
          tree

Out[137]: <ElementTree at
          0x687e970>

In [140]: root = tree.getroot()
          root

Out[140]: <Element 'kanjidic2' at
          0x0A8ABBC0>

In [143]: root.findall('character/literal')[:10]

Out[143]: [<Element 'literal' at
          0x0A8ABDB8>,
           <Element 'literal' at
          0x0A8AB368>,
           <Element 'literal' at
          0x0A8A6FC8>,
           <Element 'literal' at
          0x0A8A69C8>,
           <Element 'literal' at
          0x0A8A6308>,
           <Element 'literal' at
          0x0A8A18D8>,
           <Element 'literal' at
          0x0A89DE00>,
           <Element 'literal' at
          0x0A89D9F8>,
           <Element 'literal' at
          0x0A89D740>,
           <Element 'literal' at
          0x0A89D0E0>]
```

I *understand* now: you have to specify the exact branching in the findall command while iter works because it filters the depth first search.

Searching for the entry of a specific kanji.

```
In [144]: search_kanji = u'本'
          literals = root.findall('character/literal')
          literals[:10]
```

```
Out[144]: [<Element 'literal' at
          0x0A8ABDB8>,
           <Element 'literal' at
          0x0A8AB368>,
           <Element 'literal' at
          0x0A8A6FC8>,
           <Element 'literal' at
          0x0A8A69C8>,
           <Element 'literal' at
          0x0A8A6308>,
           <Element 'literal' at
          0x0A8A18D8>,
           <Element 'literal' at
          0x0A89DE00>,
           <Element 'literal' at
          0x0A89D9F8>,
           <Element 'literal' at
          0x0A89D740>,
           <Element 'literal' at
          0x0A89D0E0>]
```

```
In [147]: tree.find('character/literal')
```

```
Out[147]: <Element 'literal' at
          0x0A8ABDB8>
```

```
In [148]: [literal.text for literal in literals].index(u'話')
```

```
Out[148]: 2948
```

```
In [149]: print literals[2948].text
```

```
話
```

Getting the parent node.

```
In [151]: characters = root.findall('character')
          characters[:10]
```

```
Out[151]: [<Element 'character' at
          0x0A8ABA10>,
           <Element 'character' at
          0x0A8AB3E0>,
           <Element 'character' at
          0x0A8A6F98>,
           <Element 'character' at
          0x0A8A6C50>,
           <Element 'character' at
          0x0A8A6350>,
           <Element 'character' at
          0x0A8A18F0>,
           <Element 'character' at
          0x0A89DE90>,
           <Element 'character' at
          0x0A89DA70>,
           <Element 'character' at
          0x0A89D758>,
           <Element 'character' at
          0x0A89D170>]
```

```
In [153]: print characters[2948][0].text
```

```
話
```

Defining helper functions

Find a specific kanji in the dictionary

```
In [155]: def find_element_by_kanji(tree, kanji):
              root = tree.getroot()
              literals = root.findall('character/literal')
              index = [literal.text for literal in literals].index(kanji)
              return root.findall('character')[index]
```

```
In [158]: kuruma = find_element_by_kanji(tree, u'車')
          kuruma
```

```
Out[158]: <Element 'character' at
          0x0738E500>
```

```
In [160]: print kuruma[0].text
```

```
車
```

Extract meaningful information from a 'character'

```
In [173]: def extract_data(element):
              """returns the kanji, the kana and the meanings from an element"""
              kanji = element.find('literal').text
              kana = [elem.text for elem in filter(lambda reading: reading.attrib['r_type']
              meanings = [elem.text for elem in filter(lambda elem: elem.attrib == {}, elem
              return (kanji, kana, meanings)
```

```
In [176]: def disp_data(data):
              print data[0]
              for item in data[1]:
                  print item
              for item in data[2]:
                  print item

          data = extract_data(kuruma)
          disp_data(data)
```

```
車
シャ
くるま
car
```

```
In [178]: disp_data(extract_data(find_element_by_kanji(tree, u'話')))
```

```
話
ワ
はな.す
はなし
tale
talk
```

```
In [179]: disp_data(extract_data(find_element_by_kanji(tree, u'尖')))
```

```
尖
セン
とが.る
さき
するど.い
be pointed
sharp
taper
displeased
angry
edgy
```

JMdict

Working with the example file

```
In [195]: tree = ET.ElementTree(file='JMdict_example.xml')
          tree
```

```
Out[195]: <ElementTree at
          0x36692dd0>
```

```
In [196]: root = tree.getroot()
          root
```

```
Out[196]:  <Element 'entry' at
           0x366944E8>
```

Looking at the first few lines.

```
In [197]: elems = [elem for elem in tree.iter()][:10]
          elems
```

```
Out[197]:  [<Element 'entry' at
           0x366944E8>,
            <Element 'ent_seq' at
           0x36694728>,
            <Element 'k_ele' at
           0x36694500>,
            <Element 'keb' at 0x366946E0>,
            <Element 'ke_pri' at
           0x36694650>,
            <Element 'ke_pri' at
           0x36694548>,
            <Element 'ke_pri' at
           0x36694608>,
            <Element 'r_ele' at
           0x36694518>,
            <Element 'reb' at 0x36694578>,
            <Element 're_pri' at
           0x366947A0>]
```

```
In [198]: expression = root.find('k_ele/keb').text
          print expression
```

```
右翼
```

```
In [200]: reading = root.find('r_ele/reb').text
          print reading
```

```
うよく
```

```
In [203]: senses = root.findall('sense/gloss')
          senses
```

```
Out[203]:  [<Element 'gloss' at
           0x366AF2D8>,
            <Element 'gloss' at
           0x366AF320>,
            <Element 'gloss' at
           0x366AF098>,
            <Element 'gloss' at
           0x366AF368>,
            <Element 'gloss' at
           0x366AF380>,
            <Element 'gloss' at
           0x366AF3B0>,
            <Element 'gloss' at
           0x366AF410>,
            <Element 'gloss' at
           0x366AF428>,
            <Element 'gloss' at
           0x366AF440>,
            <Element 'gloss' at
           0x366AF470>,
            <Element 'gloss' at
           0x366AF4A0>,
            <Element 'gloss' at
           0x366AF4D0>]
```

```
In [208]: senses = filter(lambda sense: sense.attrib == {}, senses)
          senses
```

```
Out[208]:  [<Element 'gloss' at
           0x366AF2D8>,
            <Element 'gloss' at
           0x366AF410>,
            <Element 'gloss' at
           0x366AF428>,
            <Element 'gloss' at
           0x366AF440>]
```

```
In [209]: for sense in senses:
              print sense.text
```

```
right-wing
right field (e.g. in sport)
right flank
right wing
```

Working with the whole file

```
In [210]: tree = ET.ElementTree(file='JMdict.xml')
          tree
```

```
Out[210]:  <ElementTree at
           0x366add10>
```

In [214]:
```
root = tree.getroot()
root
```

Out[214]:
```
<Element 'JMdict' at
0x366AFE00>
```

In [217]:
```
word_entries = tree.getroot().findall('entry/k_ele/keb')
words = [entry.text for entry in word_entries]
```

In [221]:
```
for word in words[:50]:
    print word
```

```
〃
仝
々
漢数字ゼロ
○
〇
ＡＢＣ順
ＣＤプレーヤー
ＣＤプレイヤー
Ｎ響
Ｏバック
ＲＳ２３２ケーブル
Ｔシャツ
Ｔバック
あうんの呼吸
阿吽の呼吸
明白
明白
偸閑
白地
明かん
悪どい
論う
馬酔木
彼処
彼所
あっと言う間に
あっという間に
あっとゆう間に
彼の
あの人
彼の人
あの方
彼の方
溢れる
阿呆陀羅
甘子
天魚
雨子
鯇
彼
いい加減にしなさい
いい年をして
否々
否否
如何わしい
いかなる場合でも
如何にも
幾つも
行けない
```

```
In [237]: words[49][0] in words[34]

Out[237]:  False
```

Ask for a specific kanji in an expression:

In [240]:

```
filtered_words = filter(lambda expression: u'寺' in expression, words)
for word in filtered_words:
    print word
```

```
駆け込み寺
駆込み寺
古社寺
山寺
寺
寺院
禅寺
僧寺
大寺院
中禅寺湖
尼寺
仏寺
末寺
古寺
寺社
社寺
国分寺
寺参り
寺子屋
寺小屋
回教寺院
縁切り寺
氏寺
檀那寺
勅願寺
寺男
寺銭
菩提寺
寺格
八百八寺
寺内
入寺
敵は本能寺にあり
敵は本能寺に在り
寺号
寺域
官寺
大覚寺統
脇寺
寺中
寺社奉行
寺預け
寺入り
南都七大寺
七大寺
本願寺派
仏光寺派
誠照寺派
少林寺拳法
寺
お寺
御寺
お寺様
お寺さま
御寺様
紅妙蓮寺
寺請
寺請け
寺請制度
寺檀制度
三井寺歩行虫
三井寺芥虫
```

Outline of what could be done from this

- build a sort of exploratory app that starts with a kanji or a word, then lists all compounds from the dictionary that contain the given kanjis and makes it able to reselect any one of them at a later stage while offering the possibility to visualize the data associated to each kanji
- probably the most easy thing to do is classify words with respect to frequency
- add support for reading Anki decks or better: integrate with Anki desktop, as it is written in Python

```
In [ ]:
```