TEIハンズオン

応用編 「走れメロス」でテキスト分析

このセクションについて

TEIハンズオン基礎編までの知識を前提に、
TEI形式に変換された研究対象のテキスト
を使った分析例をいくつか紹介します。

demo/TEI-tools(Github)は本動画の案内から入手できます。

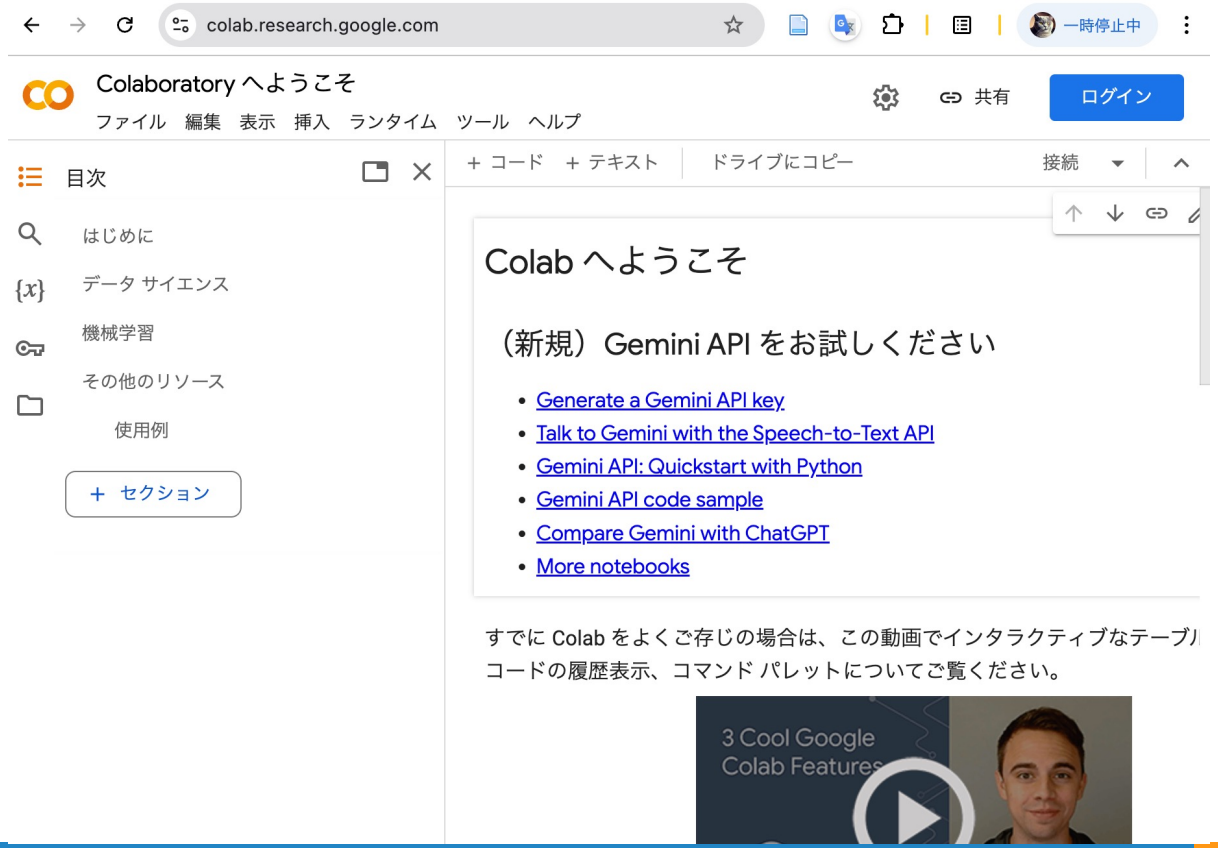
準備するもの

- demo(TEI-tools) > **TEI_hands-on.ipynb**
- Google Chromeブラウザ(推奨)

Google Colab(Colaboratory, グーグルコラブ)
https://colab.research.google.com/

Google Colab（Colaboratory）の外観

https://colab.research.google.com/?hl=ja



Google Colab（Colaboratory）?

ウェブブラウザ上で使えるお手軽プログラミングツール

- パソコンのOS（オペレーティングシステム、MacOSやWindows等）の違いや性能を気にする必要がなくなる
- 説明文をコードの前後に入力したり、トピックのアウトラインを作ることができる
- プログラミングの作成と実行の仕方がわかりやすい

https://colab.research.google.com/?hl=ja

- インターネットに接続しGoogleアカウントを作る必要がある
- プログラムを実行するには、Googleのコンピュータに接続するが長時間接続することができない
- 接続が切断されると一時的にインストールしたライブラリやデータは消える

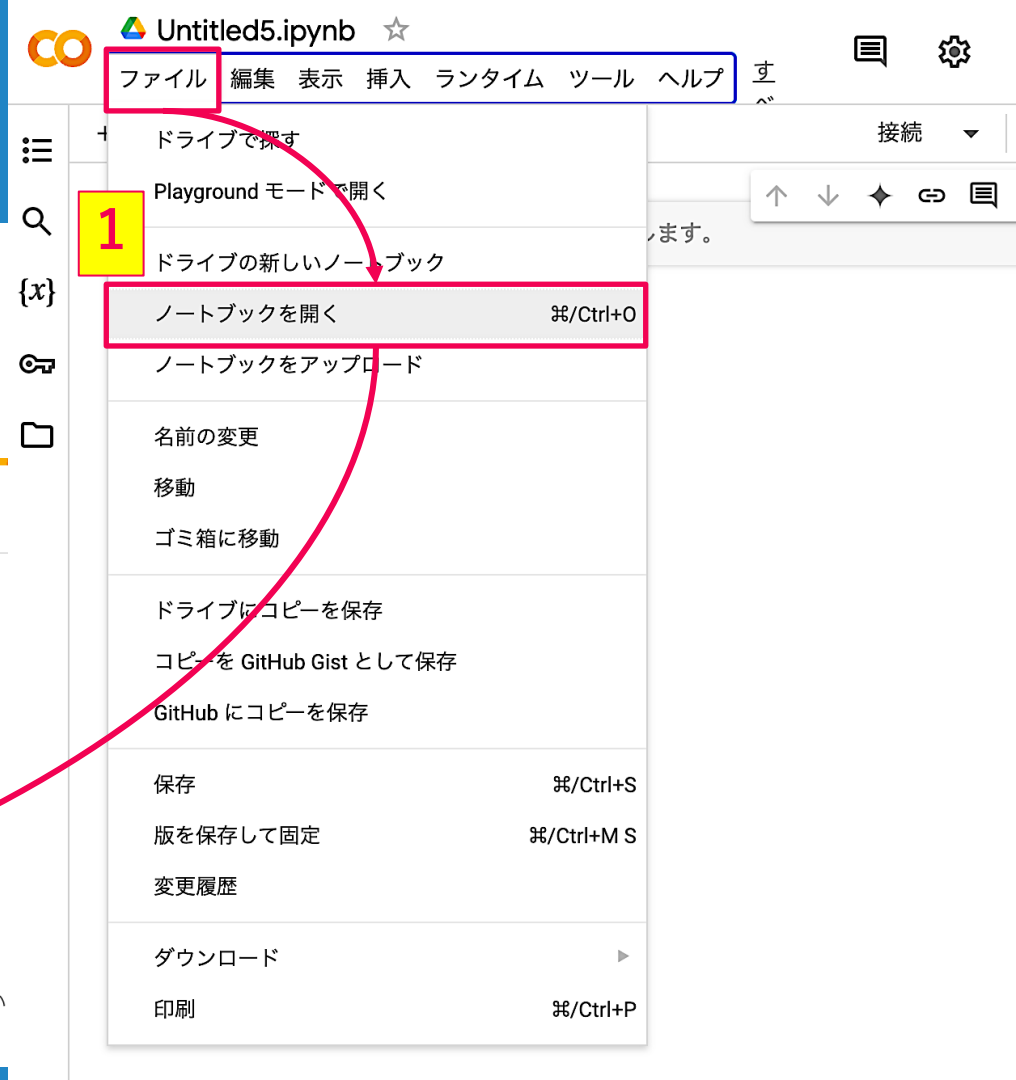
Googleアカウントを作る

- Google Colabのサービスを使うにはGoogleの共通アカウント(ユーザ名とパスワードのセット)を作る必要があります。
- アカウントを作ってColabのプログラムを動かすだけであれば無料で使えます。
- アカウントがない場合は、次の動画に進む前に以下のページでご自分のアカウントを作ってください。
  - https://www.google.com/intl/ja/account/about/ (または「Googleアカウント作成」といったキーワードでインターネットを検索してください。)

Google Colabにログインしよう

ノートブックを開こう

ファイル > ノートブックをアップロード > アップロードを選んでダウンロードした”TEI_hands-on.ipynb”を選択してください。このファイルはCLEから他のサンプルファイルとともにダウンロードできます。



TEI_hands-on.ipynbについて

Google Colabで実際に動かせる、解説付きサンプルコード

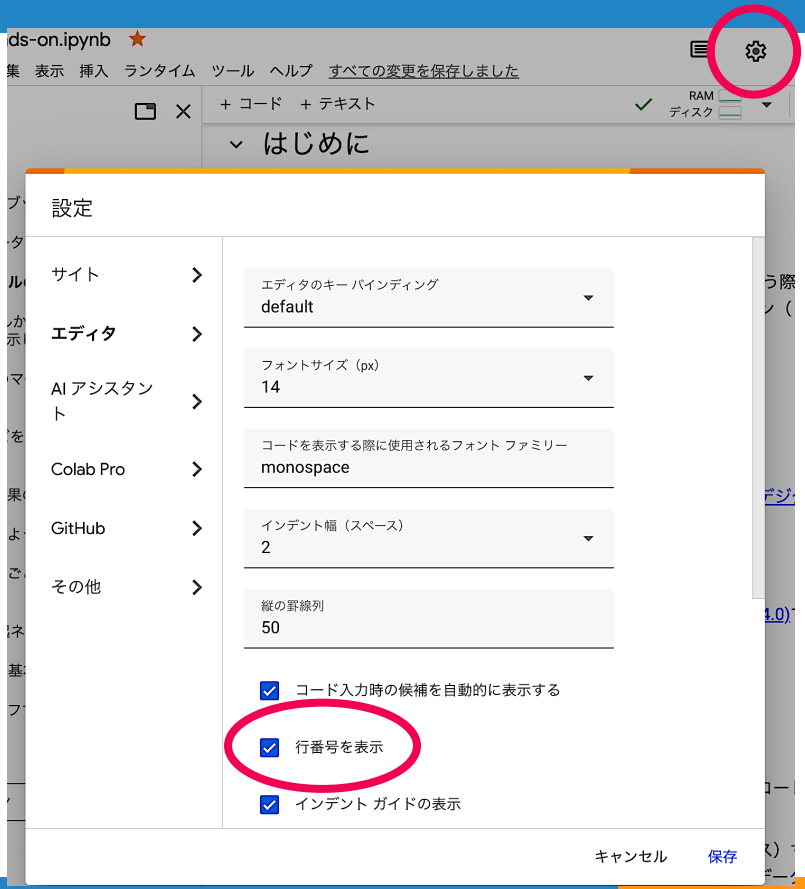
https://colab.research.google.com/drive/1t0-N22TQzqjZ-cAVtFNKbb6hcZdQ7gbX?usp=sharing

自分でコードを書き換える場合は、一旦コピーを保存してください。

サンプルコードはたくさんの事例があるPython言語で書かれています

Google Colabの設定

行番号で説明できるように歯車マークの設定から、「エディタ」をクリックして、「行番号を表示」にチェックを入れてください。



太宰治「走れメロス」のmeros.xml

青空文庫のテキストを基にTEI化したサンプルに軽微な修正を行ったもの

（底本はmeros.xmlに記載）

青空文庫有志がテキスト化

TEI-EAJ
aozora_tei
@Github

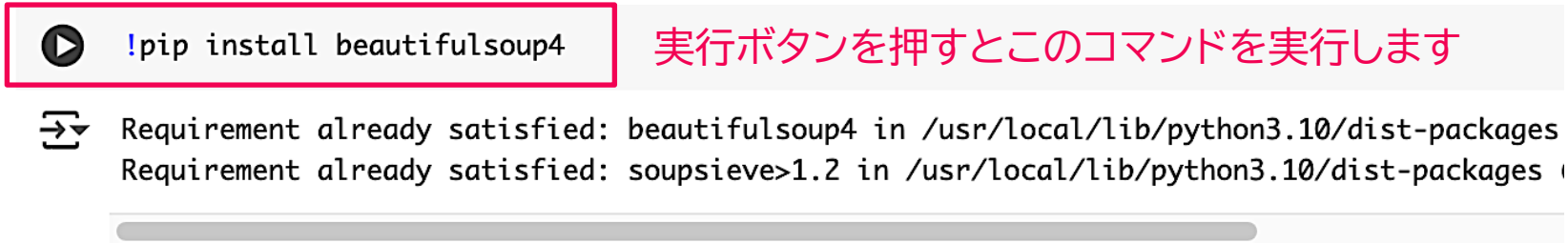
TEI-C東アジア/日本語分科会有志がTEI形式に変換

TEI

meros.xml
（イマココ）

環境設定とデータの読み込み

BeautifulSoup: PythonでXMLファイルを簡単に解析するためのライブラリ(まとまったプログラムのセット)のひとつ。任意のタグがついたデータを指定して抽出できる。



meros.xmlをダウンロード

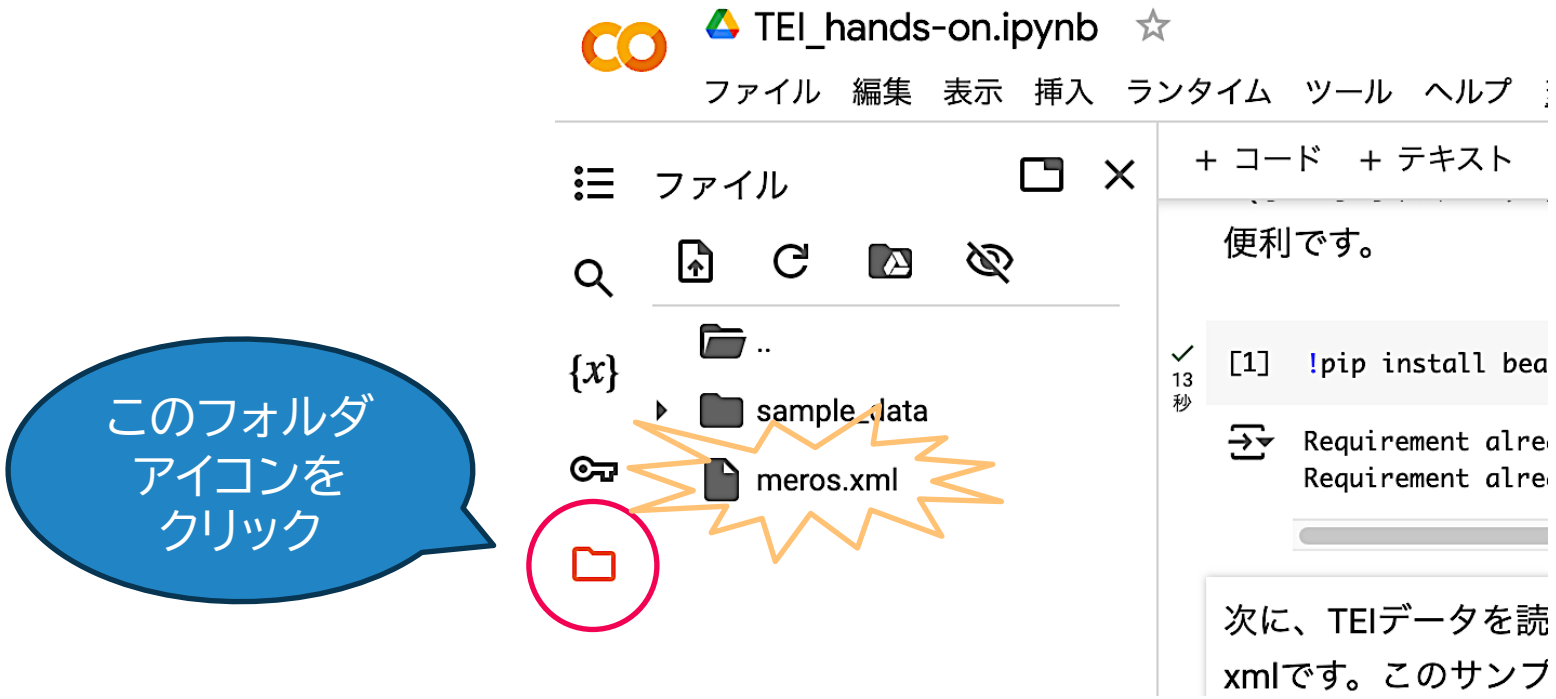
TEIデータをGithubというリポジトリ(基本無料のデータ格納庫)からダウンロードします。

押す→ 

```
!curl -O 'https://raw.githubusercontent.com/nikolito/TEI-tools/main/meros.xml'
```

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 87937  100 87937    0     0   277k      0 --:--:-- --:--:-- --:--:--  277k
```

ダウンロードしたファイルはどこへ？



meros.xmlの中身

```
with open("meros.xml", encoding='utf8') as f:
    print(f.read())
```

```
<?xml version="1.0" encoding="UTF-8"?>
<TEI xmlns="http://www.tei-c.org/ns/1.0">
      <teiHeader>
            <fileDesc>
                  <titleStmt>
                        <title>走れメロス</title>
                        <author>太宰治</author>
                        <respStmt>
                              <resp>Transcription</resp>
                              <name>金川一之</name>
                        </respStmt>
                        <respStmt>
                              <resp>Proofreading</resp>
                              <name>高橋美奈子</name>
                        </respStmt>
                        <respStmt>
                              <resp>TEI encoding</resp>
                              <name>永崎研宣</name>
                        </respStmt>
                  </titleStmt>
                  <publicationStmt>
                        <distributor>青空文庫</distributor>
                        <authority>金川一之</authority>
                        <authority>高橋美奈子</authority>
                        <date when="2011-01-17"> 2011年1月17日</date>
                  </publicationStmt>
                  <sourceDesc>
                        <bibl>
                              <author>太宰治</author>
                              <title>走れメロス</title>
```

Pythonでよく使われるプログラムパターン：
ファイルの入出力

```
with open("meros.xml",
encoding='utf8') as f:
print(f.read())
```

出力結果の表示/非表示の切り替え

このマークから
選択できます→

TEIファイルからタグを除去したテキストを表示 1

- file_path、soupなどの文字列は**変数**
  - 変数はデータに文字列の名前がついた**一時的な入れ物**
- =の右側にある値をデータとして入れる
- **関数**は何らかの値をカッコ内に入れると、値に応じた結果を出力するもの

```
[ ]  1  #モジュールのインストール
     2  from bs4 import BeautifulSoup
     3
     4  # TEIファイルを読み込んで、beautifulsoupがxml要素を取り出す
     5  file_path = "meros.xml"
     6
     7  with open(file_path, "r", encoding="utf-8") as file:
     8      soup = BeautifulSoup(file, "xml")
     9
    10  # <rt>タグを削除する
    11  for ruby in soup.find_all("rt"):
    12      ruby.decompose()
    13
    14  # <text>要素を取り出して、その中のタグを除去（タグなしの本文テキストにする）
    15  text_content = soup.find("text").get_text(strip=True)
    16
    17  # 最初の1000文字をタグなしテキストから表示する
    18  print(text_content[:1000] + "...")
```

変数

関数

TEIファイルからタグを除去したテキストを表示 2

- ルビを表示するタグのうち、rt要素を**全て抽出する関数:**
  find_all(“rt”)

ルビの例)
<ruby>
　<rb>邪智暴虐</rb>
　<rt>じゃちぼうぎゃく</rt>
</ruby>

冒頭文「メ ロ ス は 激 怒 し た 。」
0 1 2 3 4 5 6 7 8

変数の中に入っているテキストデータには0から番号が振られている

```
[ ]  1  #モジュールのインストール
     2  from bs4 import BeautifulSoup
     3
     4  # TEIファイルを読み込んで、beautifulsoupがxml要素を取り出す
     5  file_path = "meros.xml"
     6
     7  with open(file_path, "r", encoding="utf-8") as file:
     8      soup = BeautifulSoup(file, "xml")
     9
    10  # <rt>タグを削除する
    11  for ruby in soup.find_all("rt"):
    12      ruby.decompose()
    13
    14  # <text>要素を取り出して、その中のタグを除去（タグなしの本文テキストにする）
    15  text_content = soup.find("text").get_text(strip=True)
    16
    17  # 最初の1000文字をタグなしテキストから表示する
    18  print(text_content[:1000] + "...")
```

textwrapライブラリを使ってもう少し読みやすく整形

```
1  # 20文字で折り返し、300字まで表示してみる
2  import textwrap
3
4  wrapped_text = textwrap.fill(text_content, width=20)
5  print(wrapped_text[:300] + "...")
6
```

スライス記法

0番目から299番目までの文字列（=300文字分）のwrapped_text

textwrapのライブラリに含まれる**fill関数**を使ってtext_contentを幅20文字かつ300文字まで折り返す

```
メロスは激怒した。必ず、かの邪智暴虐の王
を除かなければならぬと決意した。メロスに
は政治がわからぬ。メロスは、村の牧人であ
る。
笛を吹き、羊と遊んで暮して来た。けれども
邪悪に対しては、人一倍に敏感であった。き
ょう未明メロスは村を出発し、野を越え山越
え、十里はなれた此のシラクスの市にやって
来た。メロスには父も、母も無い。女房も無
い。 十六の、内気な妹と二人暮しだ。この
妹は、村の或る律気な一牧人を、近々、花婿
として迎える事になっていた。結婚式も間近
かなのである。メロスは、それゆえ、花嫁の
衣裳やら祝宴の御馳走やらを買いに、はるば
る市にやって来たのだ。 先ず、その品々を
買い集...
```

このファイルのマークアップルールを確認するには

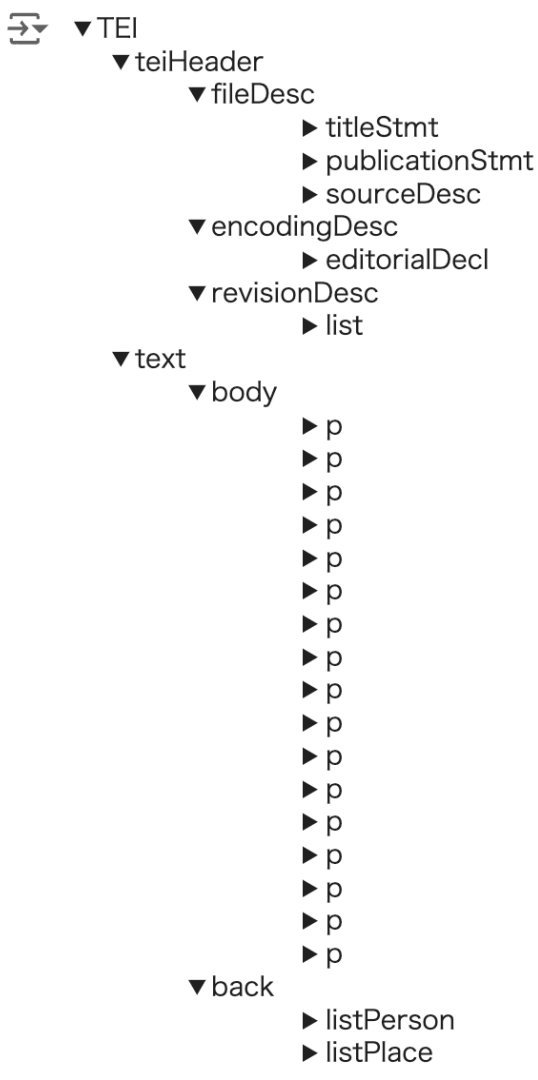
- どんなタグが使われているか？
- タグの階層構造はどうなっているのか？

```
from bs4 import BeautifulSoup
from IPython.display import display, HTML

def generate_collapsible_html(tag, indent=0):
    # 再起的にxmlタグを階層表示します
    children = tag.find_all(recursive=False)
    if not children:
        return f'<div style="margin-left: {indent}px;">{tag.name}</div>'

    inner_html = "".join([generate_collapsible_html(child, indent + 20) for child in children])
    return f"""
    <details style="margin-left: {indent}px;">
        <summary>{tag.name}</summary>
        {inner_html}
    </details>
    """

# XML要素をsoupに入れます
file_path = "meros.xml"
with open(file_path, "r", encoding="utf-8") as file:
    soup = BeautifulSoup(file, "xml")

# HTML表示でタグの階層を開閉します
root = soup.find(True)  # 最初のタグを取得
html_output = generate_collapsible_html(root)
display(HTML(html_output))
```

meros.xmlのタグ階層構造

XML形式のデータは必ず**入れ子**になっている

1. これまでの説明の通り、TEIタグの中にはteiHeader、textタグがある
2. teiHeaderにはfileDesc、encodingDesc、revisionDescタグがある
3. textタグにはbody（本文）およびback（後付けや補遺など本文以外の要素）がある
   - bodyにはpタグが17個ある
   - backにはlistPersonおよびlistPlaceタグがあり、それぞれ物語に登場する人物や地名の名称が入っている



editorialDeclタグの調査

editorialDeclとは:

マークアップルールを記すタグ

出力結果

1. placeNameの@typeでは、実際にいた場所を”real”、そうでない場所を”unreal”として区別している
2. persNameは、人称代名詞以外の人や人々を指す名詞に付与している
3. 人称代名詞にはrsを付与している
4. (上記いずれのタグも)@correspでID参照している

```
#モジュールのインストール
from bs4 import BeautifulSoup
import textwrap

# TEIファイルを読み込んで、beautifulsoupがxml要素を取り出す
file_path = "meros.xml"
with open(file_path, "r", encoding="utf-8") as file:
    soup = BeautifulSoup(file, "xml")

# マークアップルールを記した<editorialDecl>を選択して、変数editoricalDeclに代入
editorialDecl = soup.select('editorialDecl')

# editorialDeclの中身を出力する
wrapped_text = textwrap.fill(editorialDecl[0].get_text(strip=True), width=50) # widthでお好みの幅に変えてください
print(wrapped_text.replace("  ", ""))

```

名寄せをしよう

```python
from bs4 import BeautifulSoup
from collections import Counter

# TEIファイルを読み込んで、BeautifulSoupで解析
file_path = "meros.xml"  # 実際のパスを指定
with open(file_path, "r", encoding="utf-8") as file:
    soup = BeautifulSoup(file, "xml")

# <persName>タグのうちcorresp="#メロス"に該当する要素を抽出
melos_names = [pers.text for pers in soup.find_all("persName", {"corresp": "#メロス"})]

# 呼び名とその出現回数をカウント
name_counts = Counter(melos_names)

# 結果を表示
for name, count in name_counts.items():
    print(f"{name}: {count}回")
```

```
メロス: 68回
```

結果→

おかしい箇所

メロス：6回

下賤
げせん
の者：1回
嘘つき：1回
兄：4回
メロス ほどの男：1回
おまえの兄：1回
たぶん偉い男：1回
真の勇者、メロス：1回

自分：1回

よくよく不幸な男：1回
裏切者：1回
地上で最も、不名誉の人種：1回
悪徳者：1回
醜い裏切り者：1回
正義の士：1回
正直な男：2回
勇者：1回

名寄せをしよう（改良版）

```
[ ]
from bs4 import BeautifulSoup
from collections import Counter

# TEIファイルを読み込んで、BeautifulSoupで解析
file_path = "meros.xml"
with open(file_path, "r", encoding="utf-8") as file:
    soup = BeautifulSoup(file, "xml")

# <persName>タグのうちcorresp="#メロス"に該当する要素を抽出
melos_names = []

for pers in soup.find_all("persName", {"corresp": "#メロス"}):
    # <rb>タグがある場合はその内容を抽出、ない場合は全体のテキストを使用
    if pers.find("rb"):
        melos_names.append(''.join(rb.text for rb in pers.find_all("rb")).strip())
    else:
        melos_names.append(pers.text.strip())

# 呼び名とその出現回数をカウント
name_counts = Counter(melos_names)

# 結果を表示
for name, count in name_counts.items():
    print(f"{name}: {count}回")
```

ルビの例）

```
<ruby>
  <rb>邪智暴虐</rb>
  <rt>じゃちぼうぎゃく</rt>
</ruby>
```

```
メロス: 74回
下賤: 1回
嘘つき: 1回
兄: 4回
メロス ほどの男: 1回
おまえの兄: 1回
たぶん偉い男: 1回
真の勇者、メロス: 1回
自分: 1回
よくよく不幸な男: 1回
裏切者: 1回
地上で最も、不名誉の人種: 1回
悪徳者: 1回
醜い裏切り者: 1回
正義の士: 1回
正直な男: 2回
勇者: 1回
```

パラグラフごとにメロスの呼称を観察しよう

```python
from bs4 import BeautifulSoup
from collections import defaultdict
from IPython.display import display, Markdown

# TEIファイルを読み込んで、BeautifulSoupで解析
file_path = "meros.xml"  # 実際のパスを指定
with open(file_path, "r", encoding="utf-8") as file:
    soup = BeautifulSoup(file, "xml")

# text > body > pタグについて、各<p>タグごとにメロスの呼び名を抽出する
paragraphs = soup.find("text").find("body").find_all("p")
melos_names_by_paragraph = defaultdict(list)

for i, p in enumerate(paragraphs, start=1):
    for pers in p.find_all("persName", {"corresp": "#メロス"}):
        if pers.find("rb"):
            name = ''.join(rb.text for rb in pers.find_all("rb")).strip()
        else:
            name = pers.text.strip()
        melos_names_by_paragraph[i].append(name)

# 結果を表示
for para_num, names in melos_names_by_paragraph.items():
    display(Markdown(f"### Paragraph {para_num}"))
    display(Markdown(", ".join(names) if names else "--"))
```

Paragraph 1

メロス, メロス, メロス

Paragraph 2

メロス, メロス, メロス

Paragraph 3

メロス

Paragraph 4

メロス, メロス, メロス, メロス

Paragraph 5

メロス, メロス, メロス, メロス, メロス, メロス, 下賤, メロス, メロス, 嘘つき, メロス

Paragraph 6

メロス, メロス, メロス

Paragraph 7

メロス, メロス, 兄, 兄, 兄, メロス, メロス, メロス, メロス

Paragraph 8

メロス, メロス, メロス, メロス ほどの男, おまえの兄, 兄, たぶん偉い男, メロス, メロス, メロス

Paragraph 9

メロス, メロス

Paragraph 10

メロス, メロス, メロス

Paragraph 11

メロス, メロス, メロス, メロス, メロス, メロス

Paragraph 12

メロス

Paragraph 13

メロス, メロス, 真の勇者、メロス, 自分, よくよく不幸な男, 裏切者, 地上で最も、不名誉の人種, 悪徳者, 醜い裏切り者

Paragraph 14

メロス, メロス, メロス, 正義の士, 正直な男, 正直な男

Paragraph 15

メロス, メロス, メロス

Paragraph 16

メロス, メロス, メロス, メロス, メロス, メロス, メロス

Paragraph 17

メロス, メロス, メロス, メロス, メロス, メロス, メロス, メロス, メロス, メロス, メロス, メロス, 勇者

メロス、ディオニス、セリヌンティウスの呼称をパラグラフごとに抽出

index	メロス	ディオニス	セリヌンティウス	Text (first 100 chars)
P1	メロス、メロス、メロス	邪智暴虐		メロスは激怒した。必ず、かの邪智暴虐じゃちぼうぎゃくの王を除かなければならぬと決意した。メロスには政治がわからぬ。メロスは、村の牧人である。 笛を吹き、
P2	メロス、メロス、メロス			きょう未明メロスは村を出発し、野を越え山越え、十里はなれた此このシラクスの市にやって来た。メロスには父も、母も無い。女房も無い。 十六の、内気な妹と二人暮しだ。この妹は、村の或る律気な一牧人を、近々、
P3	メロス		竹馬の友、セリヌンティウス	メロスには竹馬の友があった。セリヌンティウスである。今は此のシラクスの市で、石工をしている。 その友を、これから訪ねてみるつもりなのだ。久しく逢わなかっ
P4	メロス、メロス、メロス、メロス	王、王様、御自身、国王、王		歩いているうちにメロスは、まちの様子を怪しく思った。 ひっそりしている。もう既に日も落ちて、まちの暗いのは当りまえだが、けれども、 なんだか、夜のせいばかりでは無く、市全体が、やけに寂しい。のんきなメ
P5	メロス、メロス、メロス、メロス、メロス、メロス、下賤、メロス、メロス、嘘つき、メロス	王、ディオニス、王、暴君、王、王、暴君、王、王、暴君、わし	セリヌンティウス、無二の友人、あの友人、身代りの男、身代りの男、身代り、その身代り	メロスは、 単純な男であった。買い物を、背負ったままで、 のそのそ王城にはいって行った。たちまち彼は、巡邏じゅんらの警吏に捕縛された。調べられて、メロスの懐中からは 短剣が出て来たので、

登場人物の共起ネットワーク分析

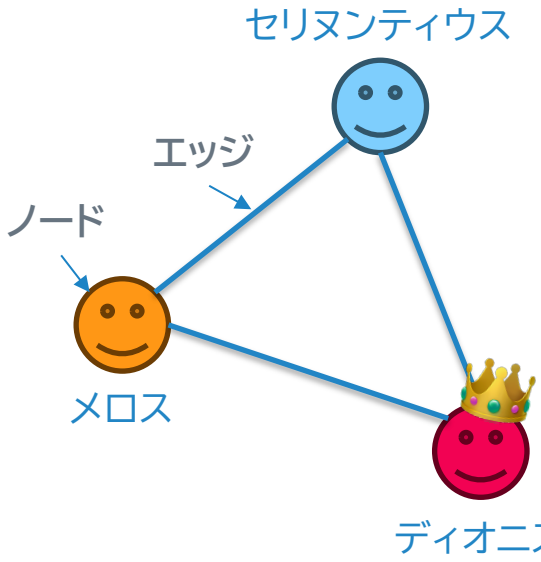
作品中で特定の人物がどの程度一緒に登場するか(共起)をパラグラフごとに分析する

手順:

1.各パラグラフで登場するキャラクター(<persName>)を抽出

2.パラグラフ単位で共に登場するキャラクター**id**のペアを集計

3.ネットワークグラフを作成し、共起頻度をエッジの太さで表現



(ネットワーク)グラフ:点(ノード)と線(エッジ)で何らかの関係を表現するデータ

登場人物の共起ネットワーク分析の準備

準備

必要なライブラリと日本語フォントを読み込む

主要ライブラリ

- **NetworkX**: Pythonでグラフ(ネットワーク)構造を扱うための強力なライブラリ。ノード(点)とエッジ(線)で構成されるネットワークを作成、操作、分析、そして可視化するために設計される。
- **matplotlib**: Pythonでデータの可視化を行うための代表的なライブラリ。特に、2Dグラフの作成に優れており、シンプルな線グラフから複雑なヒートマップや3Dプロットまで、多様なグラフ図を描画できる。

物語中の登場人物を抽出

```python
# TEIファイルを読み込んでBeautifulSoupで解析
file_path = "meros.xml"  # 実際のパスを指定
with open(file_path, "r", encoding="utf-8") as file:
    soup = BeautifulSoup(file, "xml")

# listPersonから登場人物IDと名前を抽出
all_characters = {}
for person in soup.find_all("person"):
    person_id = person.get("xml:id")
    pers_name = person.find("persName")
    if person_id and pers_name:
        name = ''.join(rb.text for rb in pers_name.find_all("rb")).strip() if pers_name.find("rb") else pers_name.text.strip()
        all_characters[person_id] = name

for character in all_characters:
  print(f"{character}: {all_characters[character]}")
```

(meros.xmlより)

```
<listPerson>
    <person xml:id="メロス">
        <persName>メロス</persName>
        <occupation>牧人</occupation>
    </person>
```

メロス: メロス
ディオニス: ディオニス
セリヌンティウス: セリヌンティウス
メロスの妹: メロスの妹
メロスの妹の婿: メロスの妹の婿
ディオニスの妹婿: 妹婿さま
ディオニスの妹: 妹さま
ディオニスの妹の御子: 御子さま
ディオニスの世嗣: お世嗣
アレキス: アレキス
老爺: 老爺
フィロストラトス: フィロストラトス

corresp属性にあるidで名寄せ

```python
# パラグラフごとの共起データを収集
paragraphs = soup.find("text").find("body").
find_all("p")
cooccurrence = defaultdict(list)

for para_num, p in enumerate(paragraphs, start=1):
    present_characters = set()

    for pers in p.find_all("persName"):
        corresp = pers.get("corresp")
        if corresp is not None:
            corresp = corresp.strip("#")
            if corresp in all_characters:
                present_characters.add(corresp)

    # 共起をパラグラフ別に記録
    for char1 in present_characters:
        for char2 in present_characters:
            if char1 != char2:
                cooccurrence[(char1, char2)].
                append(para_num)

print(f"共起ペアの数: {len(cooccurrence)}")
print("共起ペア : 共起したパラグラフ番号")
for occurrence in cooccurrence:
  print(f"{occurrence}: {cooccurrence[occurrence]}
  ")
```

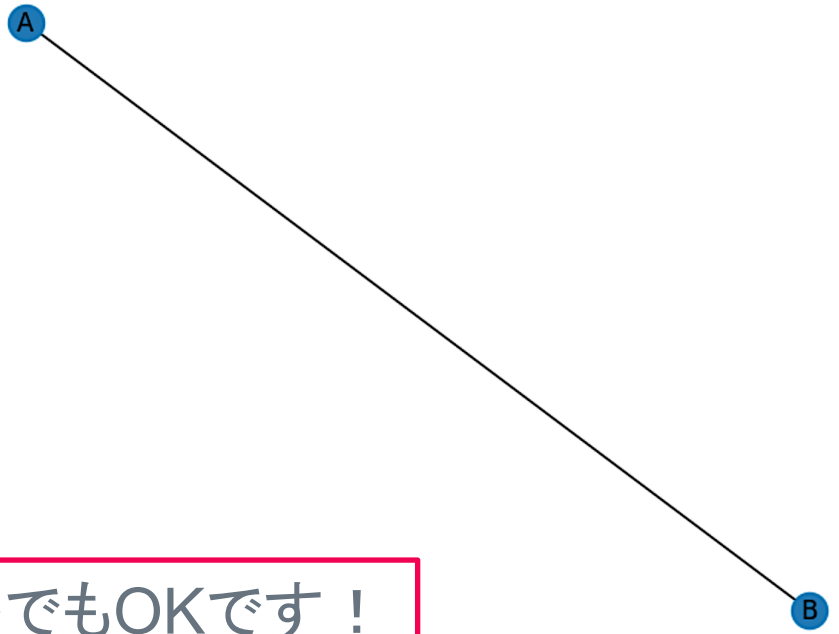
```
共起ペアの数: 78
共起ペア : 共起したパラグラフ番号
('メロス', 'ディオニス'): [1, 4, 5, 6, 8, 9, 10, 12, 13, 16, 17]
('ディオニス', 'メロス'): [1, 4, 5, 6, 8, 9, 10, 12, 13, 16, 17]
('メロス', 'メロスの妹の婿'): [2, 7, 8]
('メロス', 'メロスの妹'): [2, 5, 7, 8]
('メロスの妹の婿', 'メロス'): [2, 7, 8]
('メロスの妹の婿', 'メロスの妹'): [2, 7, 8]
('メロスの妹', 'メロス'): [2, 5, 7, 8]
('メロスの妹', 'メロスの妹の婿'): [2, 7, 8]
('メロス', 'セリヌンティウス'): [3, 5, 6, 10, 11, 13, 15, 16, 17]
('セリヌンティウス', 'メロス'): [3, 5, 6, 10, 11, 13, 15, 16, 17]
('メロス', 'ディオニスの世嗣'): [4]
('メロス', 'ディオニスの妹'): [4]
('メロス', 'ディオニスの妹の御子'): [4]
('メロス', 'アレキス'): [4]
('メロス', 'ディオニスの妹婿'): [4]
```

corresp属性を確かめれば、誰の名前なのかidでわかる

NetworkXの基本

Google Colab「TEI_hands-on.ipynb」の「NetworkXとは？」から「コードの説明」までの項目を直接参照してください。ノード間にエッジを描く基本を説明しています。

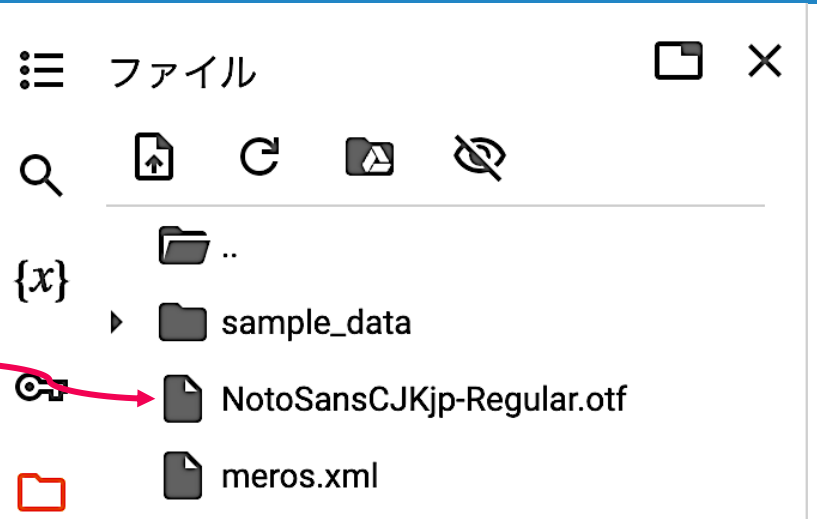
```
Nodes: ['A', 'B']
Edges: [('A', 'B', {'weight': 3})]
```



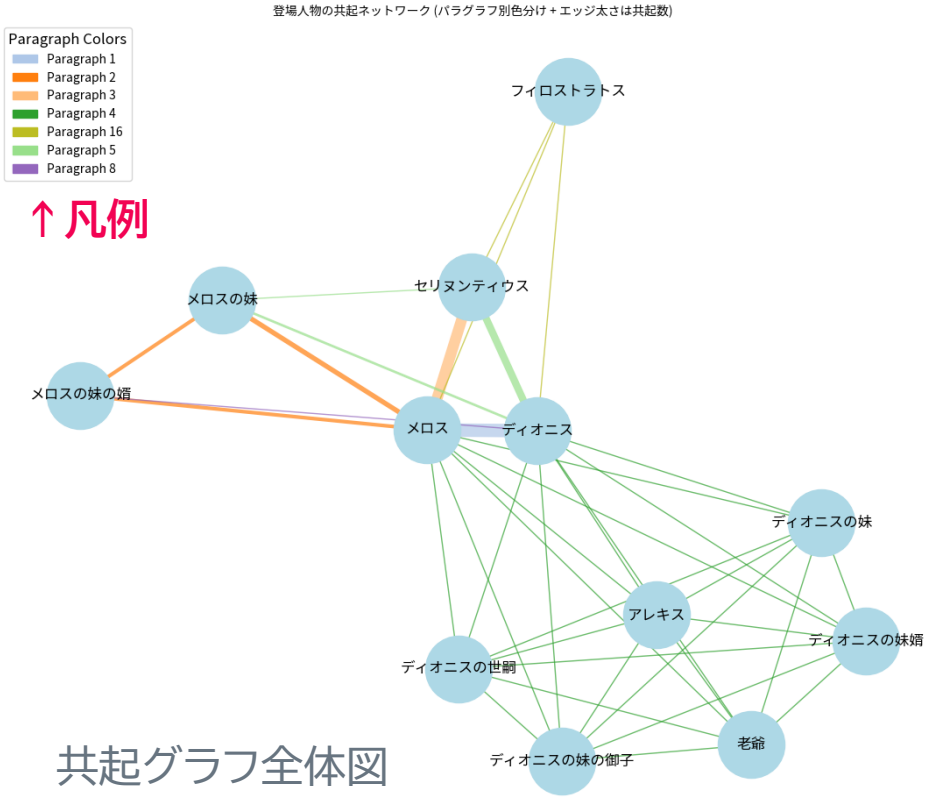
一旦次の動画に進んでもOKです！

NetworkXを使ったパラグラフ別登場人物の共起の可視化 1

```
1  import networkx as nx
2  import matplotlib.pyplot as plt
3  from bs4 import BeautifulSoup
4  from collections import defaultdict
5  import matplotlib.font_manager as fm
6  import matplotlib.patches as mpatches
7
8  # 日本語フォントの設定（Google Colabの場合には適切なフォントを利用）
9  # ColabではIPAexGothicなどが利用可能
10 ! curl -O "https://raw.githubusercontent.com/notofonts/noto-cjk/main/Sans/OTF/Japanese/NotoSansCJKjp-Regular.otf"
11
12 font_path = 'NotoSansCJKjp-Regular.otf'  # Update with your font file path
13 font_prop = fm.FontProperties(fname=font_path)
14 fm.fontManager.addfont(font_path)
15 plt.rcParams['font.family'] = font_prop.get_name()
```



NetworkXを使ったパラグラフ別登場人物の共起の可視化 2



共起グラフ全体図

【この図の読み方】

登場人物間を繋ぐエッジ

1. 太さ
2. 本数
3. 色の多さ

1-3が明確になることで、特定の登場人物同士の関係性の重要度、物語上での人物の重要度、物語の展開に伴う人間関係の変化がわかります。

より詳細な分析が必要な場合、各パラグラフの具体的な内容と照らし合わせて関係性を深掘りすることができます。

まとめ

- 「走れメロス」を使ったテキスト分析を紹介
- ノートブックはコピーして別の作品のTEIファイルに応用できる