

# 解答例

この演習では、日本の歌謡曲の歌詞データを対象に分析を行う。

使用するデータは、山本がオリコンランキングから収集した歌謡曲の歌詞に関するものである。オリコンランキングは、日本で最も知名度のある音楽ヒットランキングである。オリコンランキングの一つに年間シングル売り上げランキングというものがあり、ある年のシングル曲売り上げを集計し、そのランキングを1968年から公開し続けている。

こちらのリンクからダウンロードできるファイル `data.zip` 内には、ある年度のシングル曲売り上げ上位30件の曲の歌詞の情報が記された TSV ファイルが1968年から2024年分まで納められている（歌詞提供サイトの都合上、一部の曲が抜け落ちている）。TSV ファイル名は {順位}\_{アーティスト名}\_{曲名}.tsv の形式となっている。例えば、ダウンロードした `data.zip` ファイルを解凍して得られる `data/tokenized/2024` 年のフォルダに格納された `01_Snow Man_LOVE TRIGGER.tsv` というファイルは、「2024年のシングル売り上げ1位はSnow ManがリリースしたLOVE TRIGGERという曲」についての歌詞情報が記録されている。

また、各 TSV ファイルには、ファイル名に関する曲の歌詞を品詞単位で分割し、分割された単語に関する情報が一行ごとに格納されている。例えば、以下は `data/tokenized/2018/` フォルダにある `18_米津玄師_Lemon.tsv` ファイルの冒頭10行である。

夢	夢	名詞	一般
なら	だ	助動詞	*
ば	ば	助詞	接続助詞
どれほど	どれほど	副詞	一般
よかっ	よい	形容詞	自立
た	た	助動詞	*
でしょ	です	助動詞	*
う	う	助動詞	*
未だに	未だに	副詞	一般
あなた	あなた	名詞	代名詞

各行は歌詞に出現した単語に対応しており、出現順に並んでいる。各行のテキストは（見えないが）タブ文字 (`\t`) で区切られており、左から順に

- 活用形（歌詞中で実際に使われた形）
- 原形（辞書に載る基本形）
- 品詞（名詞・動詞など）
- 品詞詳細（助詞・代名詞などの細分類）

に関する情報が入っている。例えば、上の例では5行目の情報からは

- 米津玄師作のLemonの歌詞で5番目に出現する単語は「よかっ」で
- 品詞は「形容詞」、原形は「よい」である

ということが読み取れる。

## 準備

以下のコードを実行すると、演習で用いるファイルをダウンロードし、`data/tokenized` フォルダにデータを展開できる。なお、ダウンロードしたzipファイルを解答するにはパスワードが必要となる。教員からパスワードを入手すること。

1. `data.zip` ファイルをまだダウンロードしていない人は、以下のコードを実行してください。
2. `data.zip` ファイルを既にダウンロードした人、あるいはファイルを自力でダウンロードする人は、以下の手順をスキップしてください。

```
In [1]: import os
import requests
import zipfile

# ダウンロードURL
DATA_URL = 'https://www.ds.nagoya-cu.ac.jp/~yamamoto/download/jsong-lyrics/c

# パスワード
ZIP_PASSWORD = '*****'

# データをダウンロード
print("[STEP] ダウンロード開始...")
downloaded_data = requests.get(DATA_URL).content
with open(f'data.zip', mode='wb') as f: # wb でバイト型を書き込む
    f.write(downloaded_data)

# Zipファイル解凍
print("[STEP] 解凍中...")
zip_data = zipfile.ZipFile('data.zip', 'r')
zip_data.extractall(path='.')
zip_data.close()

# ゴミファイルの削除
print("[STEP] 余分ファイル削除中...")
for year in range(1968, 2025):
    for filename in os.listdir(f'data/tokenized/{year}'):
        if not filename.endswith('.tsv'):
            os.remove(f'data/tokenized/{year}/{filename}')

# ダウンロードしたzipファイルを削除
os.remove('data.zip')
print("[DONE] data/tokenized/ にTSVが展開されました。")
```

```
[STEP] ダウンロード開始...
[STEP] 解凍中...
[STEP] 余分ファイル削除中...
[DONE] data/tokenized/ にTSVが展開されました。
```

# 演習の進め方

上記を踏まえて、以下の課題に取り組みなさい。

課題コンテンツは以下の2種類の要素で構成されている：

- 提出課題：実データを扱う本番問題（採点対象）、全部で5つ
- 練習問題：提出課題を解くための基礎練習

練習を飛ばして提出課題に挑戦しても構いません。提出課題が難しいと感じる場合は、練習問題から順に進めるのが効果的です。

---

## 課題41

### 練習41-A: 文字列を区切る

学生に読んだ本を毎日記録し1週間ごとに報告するよう依頼した。依頼の際、データ分析の都合上、読んだ本をカンマ (,) で区切って書くようお願いした。その結果、学生Xから先週読んだ本の情報について以下のような内容が送られてきた：

コンビニ人間,ノルウェイの森,変身,ノルウェイの森,夜は短し歩けよ乙女,コンビ  
ニ人間,ノルウェイの森

今、上記内容を以下のように変数 `text` に格納したとしよう。`text` の内容をカンマで区切り、本の名前を要素とするリスト `books` を作成するコードを書きなさい。

```
text = "コンビニ人間,ノルウェイの森,変身,ノルウェイの森,夜は短し歩けよ  
乙女,コンビニ人間,ノルウェイの森"
```

関連する事項: [str.split](#)

```
In [2]: # テキストをカンマで分割してリストを作る  
text = "コンビニ人間,ノルウェイの森,変身,ノルウェイの森,夜は短し歩けよ乙女,コンビニ人間,..  
  
# split(',')でカンマ区切りのリストに変換  
books = text.split(',')  
  
# 結果の確認  
print(books)
```

```
['コンビニ人間', 'ノルウェイの森', '変身', 'ノルウェイの森', '夜は短し歩けよ乙女', 'コン  
ビニ人間', 'ノルウェイの森']
```

### 練習41-B: 辞書データの作成（リストのループ/setdefault）

以下のリストは、学生Yが1週間に読んだ本のタイトルである。それぞれの本が何回読まれたかの情報について、変数 `book_frequency` に辞書型データとして格納しなさい。

```
books = ["ハリー・ポッター", "ハリー・ポッター", "進撃の巨人", "ワンピース", "進撃の巨人", "ワンピース", "ワンピース"]
```

関連する事項: [リストのループ, dict.setdefault](#)

```
In [3]: # 学生が読んだ本のリスト
books = ["ハリー・ポッター", "ハリー・ポッター", "進撃の巨人", "ワンピース", "進撃の巨人",
         book_frequency = {} # 空の辞書を用意

# リストの中の本を順に確認
for book in books:
    # 初めて出た本なら0で初期化
    book_frequency.setdefault(book, 0)
    # 1回出てきたので+1する
    book_frequency[book] = book_frequency[book] + 1

print(book_frequency)
```

{'ハリー・ポッター': 2, '進撃の巨人': 2, 'ワンピース': 3}

### 練習41-C: 辞書のソート

練習41-Bで作成した辞書データを用いて、学生Yが読んだ本のタイトルと読書回数を回数が多い順に表示するコードを書きなさい。

関連する事項: [sorted](#)

```
In [4]: # 値(読書回数)を基準に降順で並べ替える
sorted_books = sorted(book_frequency.items(), key=lambda x: x[1], reverse=True)

# 結果を出力
for (book, count) in sorted_books:
    print(book, count)
```

ワンピース 3  
ハリー・ポッター 2  
進撃の巨人 2

## ☆課題41 (提出課題)

2019年のシングル曲売り上げ上位30件（のファイル名）を調べ、ランキング30位以内に入った回数が最も多いアーティストを表示するコードを書きなさい。

関連する事項: [os.listdir, リストのループ, str.split, dict.setdefault, sorted](#)

### ヒント

指定したフォルダ（ディレクトリ）の中にあるファイルの一覧を取得するには、`os` モジュールの `listdir` 関数を使うとよい。以下は、`data/tokenized/2024` という場所にあるフォルダ

内にあるファイル一覧を取得するコードの例である。

In [5]:

```
import os

dir_path = 'data/tokenized/2024'
filenames = os.listdir(dir_path)

print(filenames)

['22_なにわ男子_コイスルヒカリ.tsv', '13_櫻坂46_I want tomorrow to come.tsv', '19_AKB48_恋 詰んじやった.tsv', '01_Snow Man_LOVE TRIGGER.tsv', '20_AKB48_カラコンウインク.tsv', '04_Aえ! group_《A》 BEGINNING.tsv', '24_ZEROBASEONE_ゆらゆら -運命の花-.tsv', '23_Aえ! group_Gotta Be.tsv', '08_J01_WHERE DO WE GO.tsv', '18_SixTONES_GONG.tsv', '15_日向坂46_君はハニーデュー.tsv', '07_乃木坂46_チャンスは平等.tsv', '14_日向坂46_絶対的第六感.tsv', '21_SKE48_愛のホログラム.tsv', '10_SixTONES_音色.tsv', '17_SEVENTEEN_消費期限.tsv', '09_乃木坂46_チートデイ.tsv', '29_WEST_ハート.tsv', '28_&TEAM_五月雨(Samidare).tsv', '26_RIIZE_Lucky.tsv', '02_Snow Man_BREAKOUT.tsv', '27_King & Prince_halfmoon.tsv', '16_櫻坂46_何歳の頃に戻りたいのか.tsv', '05_櫻坂46_自業自得.tsv', '30_Sexy Zone_puzzle.tsv']
```

In [6]:

```
import os

# 対象年 (2019年)
dir_path = 'data/tokenized/2019'
filenames = os.listdir(dir_path)

artist_counts = {}

# 各ファイル名を調べる
for filename in filenames:
    # ファイル名は「順位_アーティスト名_曲名.tsv」
    parts = filename.split('_')
    if len(parts) >= 3:
        artist = parts[1]
        artist_counts.setdefault(artist, 0)
        artist_counts[artist] = artist_counts[artist] + 1

# 最も多く登場したアーティストを出力
sorted_artists = sorted(artist_counts.items(), key=lambda x: x[1], reverse=True)
top_artist, top_count = sorted_artists[0]
print("最も多く登場したアーティスト:", top_artist, "(", top_count, "回)")
```

最も多く登場したアーティスト: 日向坂46 ( 3 回)

---

## 課題42

### 練習42-A: リスト/辞書のループ

以下のデータは、いくつかの架空のスポーツ大会において優勝した大学の名前を年ごとにまとめたものである。ご覧の通り、変数 `results` に格納されたデータは（Pythonの）辞書型データである。

```

results = {
    "2021": ["A大学", "B大学", "A大学", "C大学"],
    "2022": ["B大学", "B大学", "C大学", "A大学"],
    "2023": ["A大学", "C大学", "C大学", "D大学"]
}

```

変数 `results` に格納されたデータを用いて、キーを「大学名」、値を「優勝回数」とする辞書型データ `frequency_win` を作成するコードを書きなさい。

関連する事項: [辞書のループ, dict.setdefault](#)

```

In [7]: results = {
    "2021": ["A大学", "B大学", "A大学", "C大学"],
    "2022": ["B大学", "B大学", "C大学", "A大学"],
    "2023": ["A大学", "C大学", "C大学", "D大学"]
}

frequency_win = {}

# 年ごとのデータを順に確認
for year in results:
    for univ in results[year]:
        frequency_win.setdefault(univ, 0)
        frequency_win[univ] = frequency_win[univ] + 1

print(frequency_win)

```

{'A大学': 4, 'B大学': 3, 'C大学': 4, 'D大学': 1}

### 練習42-B: 辞書のソート

練習42-Aで変数 `frequency_win` に格納したデータを用いて、優勝回数の下位3大学を表示するコードを書きなさい。

関連する事項: [sorted, スライスによるリストのアクセス](#)

```

In [8]: # 優勝回数が少ない順にソート
sorted_univ = sorted(frequency_win.items(), key=lambda x: x[1])
bottom3 = sorted_univ[:3]

for univ, count in bottom3:
    print(univ, count)

```

D大学 1  
B大学 3  
A大学 4

## ☆課題42（提出課題）

1968年から2024年までのシングル曲売り上げ上位30件（のファイル名）を調べ、ランキング30位以内に入った回数が多いアーティストの上位10件を表示するコードを書きなさい。

関連する事項: [range, os.listdir, str.split](#)

```
In [9]: import os

artist_counts = {}

# 1968~2024年まで繰り返す
for year in range(1968, 2025):
    dir_path = f'data/tokenized/{year}'
    filenames = os.listdir(dir_path)

    for filename in filenames:
        parts = filename.split('_')
        if len(parts) >= 3:
            artist = parts[1]
            artist_counts.setdefault(artist, 0)
            artist_counts[artist] = artist_counts[artist] + 1

# 上位10アーティストを出力
sorted_artists = sorted(artist_counts.items(), key=lambda x: x[1], reverse=True)
for artist, count in sorted_artists[:10]:
    print(artist, count)
```

AKB48 49  
嵐 38  
乃木坂46 36  
B'z 34  
SKE48 27  
KinKi Kids 23  
NMB48 20  
浜崎あゆみ 19  
松田聖子 18  
中森明菜 18

---

## 課題43

### 練習43-A: ファイルの読み込み

こちらにあるファイル (`fruit.txt`) には次のような内容が書かれている：

りんご,440  
バナナ,220  
みかん,330  
...

`fruit.txt` ファイルをダウンロードした上で、ファイルの内容を1行ずつ読み込み、果物名をキー、数字（価格）を値とする辞書データを作成しなさい。作成したデータは変数 `fruits` に格納しなさい（※ ダウンロードは手動で行えばよい）。

関連する事項: [ファイルの読み込み](#), [str.split](#)

```
In [10]: fruits = {}

# ファイルを開いて1行ずつ読む
with open('data/fruit.txt', 'r', encoding='utf-8') as f:
    for line in f:
        line = line.strip() # 文の先頭 or 末尾に付いている改行コードや空白文字を
        splitted_elements = line.split(',')
        name = splitted_elements[0]
        price = splitted_elements[1]
        fruits[name] = int(price)

print(fruits)

{'りんご': 440, 'バナナ': 220, 'みかん': 330, 'いちご': 1400, 'メロン': 1050, 'すいか': 350}
```

### 練習43-B: 辞書のキーの一覧/文字列結合

練習43-Aで作成した辞書 fruit について、その辞書のキー（つまり果物名）をカンマ（,）で結合し、文字列として出力するコードを書きなさい。

関連する事項: [dict.keys](#), [str.join](#)

```
In [11]: # dict.keys() でキー一覧を取り出す
fruit_names = fruits.keys()

# joinでカンマ区切りに結合
result = ",".join(fruit_names)

print(result)
```

りんご,バナナ,みかん,いちご,メロン,すいか

## ☆提出課題43

`data/tokenized/1976` フォルダにある `01_子門真人_およげ!たいやきくん.tsv` ファイルを1行ずつ読み込み、単語の出現順に「活用形」を取り出しつなげることで、「およげ!たいやきくん」の歌詞を復元しなさい。

関連する事項: [ファイルの読み込み](#), [str.join](#)

```
In [12]: import os

path = 'data/tokenized/1976/01_子門真人_およげ!たいやきくん.tsv'
lyrics = []

# ファイルを読み込み、1列目の活用形をつなげる
with open(path, 'r', encoding='utf-8') as f:
    for line in f:
        splitted_elements = line.strip().split('\t')
        word = splitted_elements[0]
        lyrics.append(word)

# すべての単語をつなげて歌詞にする
```

```
song = "\n".join(lyrics)
print(song)
```

毎日毎日記号僕らは鉄板の記号上で焼かれて記号嫌になっちゃうよある朝記号僕は記号店のおじさんと記号けんかして記号海に逃げこんだのさ初めて泳いだ海の底記号とっても気持ちがいいもんだお腹のあんこが重いけど記号海は広いぜ心がはずむ桃いろさんが手を振って記号僕の泳ぎを眺めていたよ毎日毎日記号楽しいことばかり記号難破船が僕のすみかさときどき鮫に記号いじめられるけど記号そんなときや記号そまさ記号逃げるのさ一日泳げばはらぺこさ記号目玉もくるくる回っちゃうたまにはえびでも食わなけりや記号塩水ばかりじゃふやけてしまう岩場のかげから食いつけば記号それは小さなつりぱりだったどんなにどんなに記号もがいても記号針が喉からとれないよ浜べで見知らぬおじさんが記号僕を釣り上げびっくりしてたやっぱり僕はたいやきさ記号少し焦げあるたいやきさおじさん記号つばをのみ込んで記号僕をうますに食べたのさ

---

## 課題44

### 練習44-A: 辞書のループ/条件分岐

変数 `animals` は、動物とその分類に関する情報が格納された辞書である。

```
animals = {
    "犬": "哺乳類",
    "カエル": "両生類",
    "トカゲ": "爬虫類",
    "猫": "哺乳類",
    "ワニ": "爬虫類"
}
```

分類が哺乳類の動物だけを抽出し、その名前（動物名）を要素とするリスト `animal_names` を作成するコードを書きなさい。

関連する事項: [dict.items](#)

```
In [13]: animals = {
    "犬": "哺乳類",
    "カエル": "両生類",
    "トカゲ": "爬虫類",
    "猫": "哺乳類",
    "ワニ": "爬虫類"
}

animal_names = []

for name, kind in animals.items():
    if kind == "哺乳類":
        animal_names.append(name)

print(animal_names)

['犬', '猫']
```

### 練習44-B: 条件つきカウント

練習44-Aで用いた変数 `animals`において、分類が「哺乳類」以外の動物の数をカウントするコードを書きなさい。

関連する事項: [dict.items](#)

```
In [14]: count = 0

for name, kind in animals.items():
    if kind != "哺乳類":
        count = count + 1

print("哺乳類以外の動物の数:", count)
```

哺乳類以外の動物の数: 3

### 練習44-C: ファイルの読み込み、平均値の計算

ダウンロードした歌詞関連データ (`data.zip`) 内にある `data/songlist.tsv` は、歌詞データセットの中で対象としている曲（つまり `tokenized` フォルダ以下で取り扱われている曲）を一覧にしたTSVファイルである。このTSVファイルの各行には以下の情報がタブ文字 ('\t': 不可視文字) 区切りで記されている。

- ランキング集計された年
- ランキング順位
- アーティスト名
- 曲名
- 感情スコア: 値が1に近いほど歌詞がポジティブ、-1に近いほど歌詞がネガティブ

`songlist.tsv` ファイルを読み込み、1968年から2024年までの感情スコアの平均値の一覧を表示するコードを書きなさい。

関連する事項: [ファイルの読み込み](#), [str.split](#), [int/float](#), [dict.setdefault](#), [リストのループ](#)

```
In [15]: scores = {}

with open('data/songlist.tsv', 'r', encoding='utf-8') as f:
    for line in f:
        splitted_elements = line.strip().split('\t')
        if len(splitted_elements) >= 5:
            year = splitted_elements[0]
            score = float(splitted_elements[4])
            scores.setdefault(year, [])
            scores[year].append(score)

# 各年の平均を出力
for year in sorted(scores.keys()):
    avg = sum(scores[year]) / len(scores[year])
    print(year, round(avg, 3))
```

1968	-0.274
1969	-0.005
1970	-0.333
1971	-0.121
1972	-0.15
1973	-0.116
1974	-0.177
1975	-0.22
1976	-0.116
1977	-0.144
1978	-0.167
1979	0.102
1980	-0.005
1981	-0.08
1982	0.197
1983	-0.064
1984	-0.066
1985	0.067
1986	0.231
1987	-0.082
1988	0.281
1989	0.014
1990	0.219
1991	0.115
1992	0.135
1993	0.14
1994	0.036
1995	0.119
1996	0.373
1997	0.365
1998	0.385
1999	0.435
2000	0.387
2001	0.307
2002	0.425
2003	0.278
2004	0.522
2005	0.432
2006	0.459
2007	0.448
2008	0.372
2009	0.223
2010	0.56
2011	0.691
2012	0.41
2013	0.411
2014	0.544
2015	0.622
2016	0.452
2017	0.439
2018	0.419
2019	0.299
2020	0.361
2021	0.306
2022	0.177

2023 0.307

2024 0.523

## ☆提出課題44

歌詞における代名詞の使用頻度を分析したい。1968年から2024年までのランキングに入っている曲の歌詞から代名詞を抽出し、その代名詞が歌詞中で用いられている曲数を辞書型で格納しなさい（代名詞をキー、曲数を値とする辞書データを作成しなさい）。また、使用頻度（使用されている曲数）が高い順に代名詞をソートし、その上位20件を表示しなさい。

\* 代名詞を抽出する際、以下の指示語は無視すること。

```
STOPWORDS = ['これ', 'それ', 'あれ', 'どれ', 'ここ', 'そこ', 'あそこ', 'どこ']
```

関連する事項: [range](#), [os.listdir](#), ファイルの読み込み, [dict.items](#), [dict.setdefault](#)

```
In [16]: import os

STOPWORDS = ['これ', 'それ', 'あれ', 'どれ', 'ここ', 'そこ', 'あそこ', 'どこ']

pronoun_counts = {}

for year in range(1968, 2025):
    dir_path = f"data/tokenized/{year}"
    filenames = os.listdir(dir_path)

    for filename in filenames:
        path = os.path.join(dir_path, filename)
        pronouns_in_song = set()

        with open(path, 'r', encoding='utf-8') as f:
            for line in f:
                splitted_elements = line.strip().split('\t')
                if len(splitted_elements) >= 3:
                    word = splitted_elements[0]
                    pos = splitted_elements[2]
                    if pos == "名詞" and splitted_elements[3] == "代名詞" and
                        pronouns_in_song.add(word)

        # その曲で出た代名詞をカウント
        for p in pronouns_in_song:
            pronoun_counts.setdefault(p, 0)
            pronoun_counts[p] = pronoun_counts[p] + 1

# 上位20件を出力
sorted_pronouns = sorted(pronoun_counts.items(), key=lambda x: x[1], reverse=True)
for word, count in sorted_pronouns[:20]:
    print(word, count)
```

君 596  
誰 493  
僕 433  
あなた 430  
何 413  
いつ 312  
私 307  
僕ら 156  
俺 134  
みんな 129  
おまえ 52  
お前 50  
わたし 49  
何処 49  
ぼく 49  
いくつ 47  
なん 41  
彼 37  
キミ 37  
あいつ 32

---

## 課題45

### 練習45-A: 辞書の辞書/ループの入れ子

変数 `sales` に格納されたデータは、年代ごとに商品A, B, Cが売れた個数を示している。各商品の合計販売数を求めるコードを書きなさい（例: Aは18個）。

```
sales = {
    1990: {"A": 10, "B": 5},
    2000: {"A": 3, "C": 8},
    2010: {"B": 7, "C": 10},
    2020: {"A": 5, "B": 2, "C": 3}
}
```

関連する事項: [dict.items](#), [dict.setdefault](#)

```
In [17]: sales = {
    1990: {"A": 10, "B": 5},
    2000: {"A": 3, "C": 8},
    2010: {"B": 7, "C": 10},
    2020: {"A": 5, "B": 2, "C": 3}
}

totals = {}

for year, items in sales.items():
    for product, num in items.items():
        totals.setdefault(product, 0)
        totals[product] = totals[product] + num
```

```
print(totals)
{'A': 18, 'B': 14, 'C': 21}
```

### 練習45-B: 辞書ソート再び

練習45-Aで用いたデータにおいて、最も販売合計数が多い商品名とその数を出力するコードを書きなさい。

関連する事項: `dict.items`, `dict.setdefault`, `sorted`

```
In [18]: # 販売数が多い順に並べる
sorted_totals = sorted(totals.items(), key=lambda x: x[1], reverse=True)

# 一番多いものを出す
top_product, top_sales = sorted_totals[0]
print("最も多く売れた商品:", top_product, " (", top_sales, "個) ")
```

最も多く売れた商品: C ( 21 個)

## ☆提出課題45

歌詞における言葉の使われ方は、社会や文化、時代ごとの流行を反映していると考えられる。そこでこの課題では、「僕」「君」といった一人称/二人称代名詞に着目したい。

以下の `PRONOUNS` に記した一人称/二人称代名詞について、注目する代名詞が出現する曲数を年代別に分析し、結果を表示しなさい。なお、「年代」のグルーピングは1960年代、1970年代、1980年代、1990年代、2000年代、2010年代、2020年代としなさい。

```
PRONOUNS = ["僕", "私", "僕ら", "俺", "わたし", "ぼく", "君", "あなた", "おまえ", "お前", "キミ"]
```

関連する事項: `os.listdir`, ファイルの読み込み, リストのループ, `dict.items/keys`, `dict.setdefault`, `in`, `sorted`

```
In [19]: import os

PRONOUNS = ["僕", "私", "僕ら", "俺", "わたし", "ぼく", "君", "あなた", "おまえ",

# 各年代ごとの辞書
decade_counts = {
    "1960年代": {},
    "1970年代": {},
    "1980年代": {},
    "1990年代": {},
    "2000年代": {},
    "2010年代": {},
    "2020年代": {}
}

for year in range(1968, 2025):
    dir_path = f"data/tokenized/{year}"
    filenames = os.listdir(dir_path)
```

```

# 年から年代ラベルを決める
# if year >= 1960 and year <= 1969:
#     decade_label = '1960年代'
# elif year >= 1970 and year <= 1979:
#     decade_label = '1970年代'
# .....

decade_label = str(year // 10 * 10) + "年代"

for filename in filenames:
    path = os.path.join(dir_path, filename)

    # ある1曲の中で出現した代名詞の数をカウントする辞書
    found_pronouns = {}

    with open(path, 'r', encoding='utf-8') as f:
        for line in f:
            splitted_elements = line.strip().split('\t')
            if len(splitted_elements) >= 1:
                word = splitted_elements[0]
                if word in PRONOUNS:
                    found_pronouns.setdefault(word, 0)
                    found_pronouns[word] += 1

    #代名詞が出現した曲数を更新する
    for p in found_pronouns:
        decade_counts[decade_label].setdefault(p, 0)
        decade_counts[decade_label][p] += 1

# 結果を見やすく出力
for decade in decade_counts:
    print("===", decade, "===")
    for pronoun, count in sorted(decade_counts[decade].items(), key=lambda x:
        print(pronoun, count)

```

==== 1960年代 ===

私 20  
あなた 19

君 10

僕 5

ぼく 4

俺 3

わたし 3

お前 2

おまえ 1

==== 1970年代 ===

あなた 133

私 94

君 46

僕 34

わたし 24

ぼく 19

俺 18

おまえ 10

お前 9

僕ら 2

==== 1980年代 ===

あなた 97

私 71

君 61

俺 45

僕 28

お前 23

おまえ 20

わたし 15

ぼく 7

僕ら 1

==== 1990年代 ===

君 107

あなた 82

僕 65

私 46

僕ら 20

俺 18

ぼく 14

おまえ 10

キミ 5

お前 4

わたし 3

==== 2000年代 ===

君 148

僕 109

僕ら 57

あなた 56

私 40

俺 25

キミ 19

お前 5

ぼく 5

おまえ 3

わたし 1

==== 2010年代 ===

君 150  
僕 122  
僕ら 46  
あなた 33  
私 28  
俺 18  
キミ 10  
おまえ 6  
お前 4  
わたし 3

==== 2020年代 ===

君 86  
僕 70  
僕ら 30  
あなた 10  
私 8  
俺 7  
キミ 4  
お前 3  
わたし 2  
おまえ 2

In [ ]: