Pythonプログラミング　中級

2023年12月16日（土）第3回
講師：渡邉 貴之

タイムテーブル

第1回 （12月9日）	**9:30 ～ 10:15**
	pandasによるデータの集計と可視化1
第2回 （12月9日）	**10:25 ～ 11:30**
	pandasによるデータの集計と可視化2
第3回 （12月16日）	**9:30 ～ 10:15**
	MeCabによる自由回答の分析
第4回 （12月16日）	**10:25 ～ 11:35**
	scikit-learnによる回帰分析

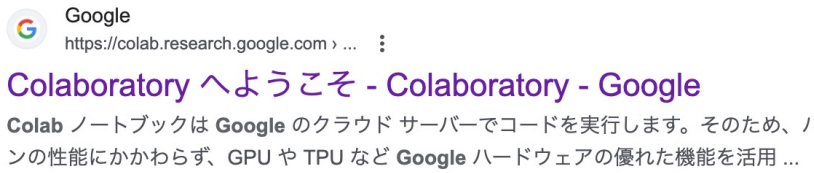
第3回
MeCabによる自由回答の分析

目次

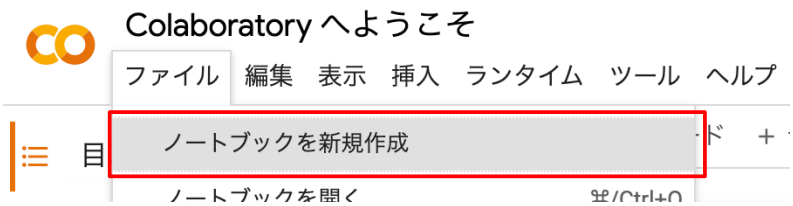
- 実習環境の準備
- 演習で使用するモジュール
- テキストマイニング
  - 形態素解析
  - 日本語形態素解析器
  - MeCabの環境設定
  - MeCabによる形態素解析の初歩
  - MeCabとUniDicによる解析結果例
  - 解析した形態素を順に取り出すには
  - 旅行サイトの口コミ(自由回答)分析
  - データのGoogle Colabへのアップロード
  - データの取り込み
  - 自由回答から単語(名詞)を取り出す
  - 単語(名詞)の件数をカウントする
  - ワードクラウドを描く
  - 出力例
- 付録
  - ストップワードの設定
  - 課題
- 初級レベルの制御構文の確認
  - 制御構文:逐次、選択、繰り返し
  - 選択(if文とelse文)
  - 条件式の書き方
  - 繰り返し(while文)
  - 繰り返し(for文)

実習環境の準備(1)

- 今回の実習では、インストール不要のPython環境であるGoogle Colaboratory(以下、Google Colab)を使用します。
  - Googleアカウントをご用意ください
  - 手順
    - Webブラウザ(Edge/Chrome/Safari等)を起動する
    - google colaboratory で検索する
    - 下記リンクを開く

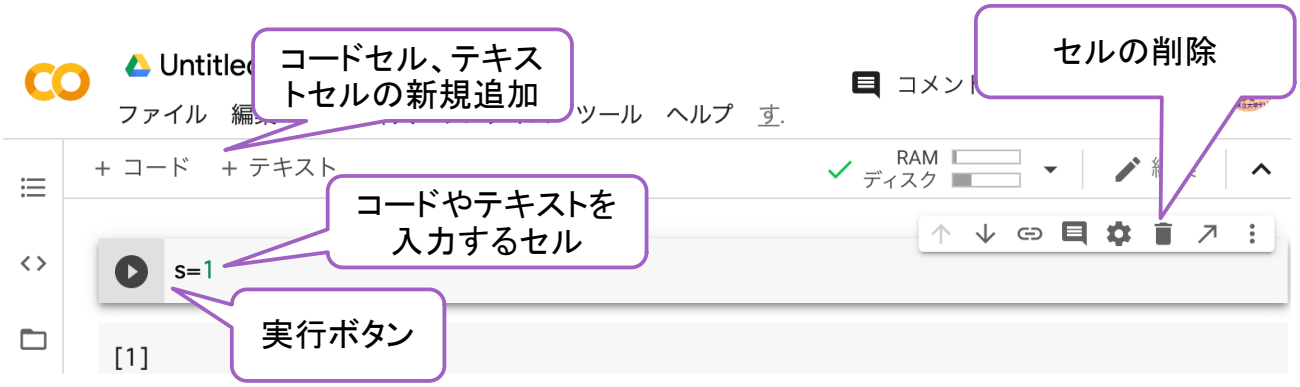


    - 「ファイル」メニューから「ノートブックを新規作成」を選択



実習環境の準備(2)

- Google Colaboratoryでは、ノートブックという単位でソースコードを記述します。
  - コードやテキストを入力する枠をセルと呼びます。
  - セルには、Pythonのコードを入力するコードセルと、説明書きを入力するテキストセルがあり、メニューからセルを追加することができます。
  - コードは実行ボタンで実行できるほか、キーボードのCtrl＋Enterキーでも実行できます。また、Shift＋Enterキーで実行とセルの追加が同時にできます。



  - ノートブックを開き直した場合は、必要なファイルを再アップロードして、一番上のセルから順に実行し直してください。

演習で使用するモジュール

- 下記の表の赤文字のモジュールを使用します。

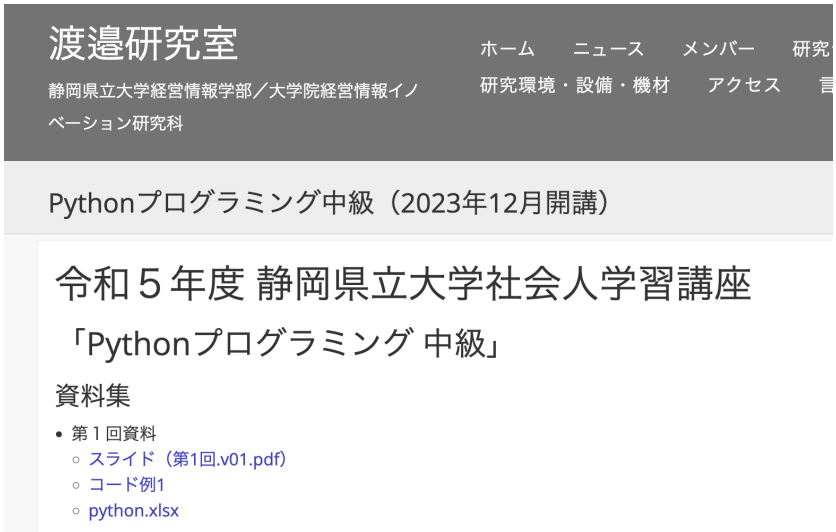
モジュール名	概要
matplotlib	グラフなどの可視化モジュール
seaborn	matplotlibの見栄えをより綺麗にするモジュール
NumPy	計算を効率的に行うためのモジュール
SymPy	数式・記号計算用モジュール
pandas	データ解析支援モジュール（Excelファイルの読み書きが可能）
openpyxl	Excelファイルの読み書きに特化したモジュール
xlwings	Excelアプリを直接制御できるモジュール
scipy	NumPyを利用した数値解析モジュール
scikit-learn	機械学習モジュール
NetworkX	グラフ・ネットワーク計算と可視化モジュール
Basemap	地図描画モジュール
MeCab	形態素解析モジュール
wordcloud	ワードクラウド可視化モジュール

資料電子データサイト

- ブラウザ（Edge/Chrome）で下記URLを開く。

  https://wtclab.tech/ppm23

  - 本日使用する教材ファイルやサンプルデータ等が掲載されています。必要に応じてダウンロードします。



テキストマイニング

テキストマイニング

- 大量のテキストデータから有益な知識や知見を見つけ出すのを助ける技術

- テキストマイニングの３要素
  - 情報の抽出
    - どのようにテキストデータを集めるか
    - 今回：トリップアドバイザーの口コミ
  - 抽出した情報の解析
    - 集めてきたテキストデータをどのように分析・解析するか
    - 今回：MeCabによる形態素解析
  - 解析結果の可視化
    - 解析結果の考察と理解を容易にするためにどのように可視化するか
    - 今回：ワードクラウドによる可視化

形態素解析

- Morphological Analysis
  - 与えられたテキストデータを形態素に分ける作業
  - 形態素とは
    - 単語に近い概念
    - 文法的に意味付けが可能な最小単位
  - 例文：
    - 静岡県立大学でpythonの講座を受講しています。

形態素	静岡県立大学	で	python	の	講座	を	受講	し	て	い	ます	。
品詞	名詞 固有名詞 一般	助詞 格助詞	名詞 固有名詞 一般	助詞 格助詞	名詞 普通名詞 一般	助詞 格助詞	名詞 普通名詞	動詞	助詞 接続助詞	動詞	助動詞	補助記号

日本語形態素解析器

- コンピュータを利用して日本語テキストデータの形態素解析を実行するプログラム／エンジン
  - MeCab
    - ChaSenを元に工藤拓氏（奈良先端大卒、現Google）によって開発されているオープンソースの形態素解析エンジン
  - ChaSen
    - JUMANを元に奈良先端大の松本研究室で開発されたオープンソースの形態素解析エンジン
  - JUMAN
    - 京都大学の黒橋・褚・村脇研究室で開発されているオープンソースの形態素解析エンジン
  - Janome
    - Python専用の形態素解析エンジン
  - ・・・

MeCabの環境設定

- Google ColabにはMeCabはインストールされていないため、!pip installコマンドでインストールします。
  - 「!」から始まるコマンドは、pythonの文法とは関係のないGoogle Colabのシステム制御用コマンドのため、細かな意味は不明で大丈夫です（おまじない）。
- MeCabの使用には「辞書」ファイルが必要となるため追加でUniDicという辞書をインストールします。
  - 辞書とは・・・「pyhon」という単語であれば、「読み：パイソン、品詞：名詞」などの情報が記述されたファイル
  - UniDic辞書：国立国語研究所が公開している形態素解析用の辞書

- 手順：
  - セルに下記のコマンドを記述して実行します。

```
#MeCabのインストール
!pip install mecab-python3
#UniDic辞書のインストールとダウンロード
!pip install unidic
!python -m unidic download
```

  - インストールとダウンロードが完了するまで待ちます。

MeCabによる形態素解析の初歩

- Taggerオブジェクトを生成して、parseメソッドで解析します。
  - 新しいコードセルを追加して、下記のコードを入力して実行します。

```
#MeCabとunidicを使用するためのインポート文
import MeCab
import unidic

#文書を形態素に分けてタグ付けする機能を持つオブジェクト（タガー）を取得する
tagger = MeCab.Tagger()
#parseメソッドで形態素解析を実行する
print(tagger.parse("静岡県立大学でpythonの講座を受講しています。"))
```

MeCabとUniDicによる解析結果例

- 表層形（surface）・・・文中に現れた語形のこと

表層形　品詞大分類　品詞細分類1　品詞細分類2・・・

```
静岡 名詞,固有名詞,地名,一般,,,シズオカ,シズオカ,静岡,シズオカ,静岡
県　名詞,普通名詞,一般,,,,ケン,県,県,ケン,県,ケン,漢,"ケ濁","基本形
立　接尾辞,名詞的,一般,,,,リツ,立,立,リツ,立,リツ,漢,"","","","","","
大学 名詞,普通名詞,一般,,,,ダイガク,大学,大学,ダイガク,大学,ダイガク
で　助詞,格助詞,,,,,デ,で,で,デ,で,デ,和,"","","","","","",格助,デ,デ
python　名詞,普通名詞,一般,,,
の　助詞,格助詞,,,,,ノ,の,の,ノ,の,ノ,和,"","","","","","",格助,ノ,ノ,
講座 名詞,普通名詞,一般,,,,コウザ,講座,講座,コーザ,講座,コーザ,漢,"
を　助詞,格助詞,,,,,ヲ,を,を,オ,を,オ,和,"","","","","","",格助,ヲ,ヲ,
受講 名詞,普通名詞,サ変可能,,,,ジュコウ,受講,受講,ジュコー,受講,ジュ
し　動詞,非自立可能,,,サ行変格,連用形-一般,スル,為る,し,シ,する,ス
て　助詞,接続助詞,,,,,テ,て,て,テ,て,テ,和,"","","","","","",接助,テ
い　動詞,非自立可能,,,上一段-ア行,連用形-一般,イル,居る,い,イ,いる
ます 助動詞,,,,助動詞-マス,終止形-一般,マス,ます,ます,マス,ます,マス
。　補助記号,句点,,,,,,。,。,,。,,記号,"","","","","","",補助,,,,,"","
EOS
```


解析した形態素を順に取り出すには

- 解析した形態素を順に取り出すには、parseメソッドの代わりにparseToNodeメソッドを使用します。
  - while文を使用してnodeが空になるまで順に取り出すことができます。

最初のnodeの内容（surfaceが空白、featureがBOS/EOS,*,*...・・）

[空白]　今日　は　とても　寒い　です　。　[空白]

```
node = tagger.parseToNode("今日はとても寒いです。")
while node:
    print(node.surface, "\t", node.feature)
    node = node.next
```

字下げ

次のnodeを新たなnodeとする

```
	BOS/EOS,*,*,*,*,*,*,*,*,*,*,*,*,*,*,*,*,*,*,*
今日	名詞,普通名詞,副詞可能,*,*,*,キョウ,今日,今日,キョー,今日,キョー,和
は	助詞,係助詞,*,*,*,*,ハ,は,は,ワ,は,ワ,和,*,*,*,*
とても	副詞,*,*,*,*,*,トテモ,迚も,とても,トテモ,とても,トテモ,和,*,*,*,*
寒い	形容詞,一般,*,*,形容詞,終止形-一般,サムイ,寒い,寒い,サムイ,寒い,サ
です	助動詞,*,*,*,助動詞-デス,終止形-一般,デス,です,です,デス,です,デ
。	補助記号,句点,*,*,*,*,,。,。,,。,,記号,*,*,*,*
	BOS/EOS,*,*,*,*,*,*,*,*,*,*,*,*,*,*,*,*,*,*,*
```

surface

feature

旅行サイトの口コミ(自由回答)分析

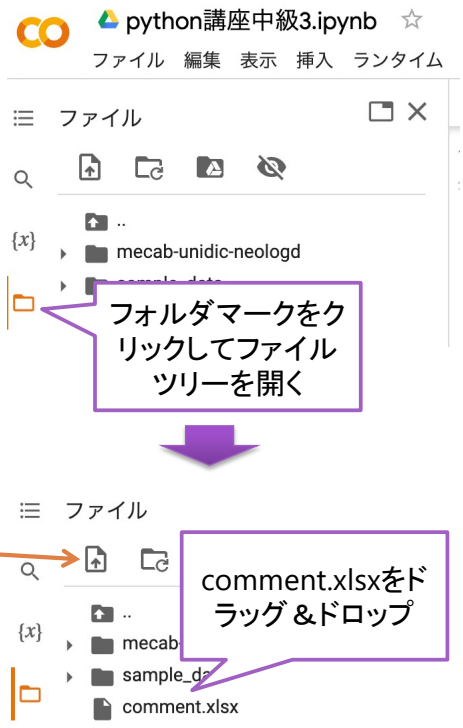
- 自由回答の分析の一例として、旅行サイト(トリップアドバイザー)の口コミ文章に含まれる、単語の出現頻度を集計してみましょう。
  - 対象観光スポット:白糸の滝(静岡県富士宮市)
  - 口コミのデータファイル:資料電子データサイトの「comment.xlsx」

	居住地	自由回答
2	愛知県	お勧めの時間帯は早朝です。富士山が世界遺産
3	愛知県	晴れていて絶景を一望できました。また足元は
4	愛知県	水の流れの清らかさマイナスイオンを感じる穏
5	愛知県	工事前、工事後にそれぞれ期間を空けて行って
6	愛知県	平日の3時頃行ったので比較的空いていました。
7	愛知県	白糸の滝と呼ばれるものは他にも軽井沢などに
8	愛知県	朝早かったせいなのか、有料駐車場に無料で駐

データのGoogle Colabへのアップロード

- comment.xlsxをGoogle Colabにアップロードして、読み込めるようにします。

- 手順
  - 資料電子データサイトから、comment.xlsxをダウンロードする
  - Google Colab左のフォルダマークをクリックする
  - ファイルツリーが開くので、ダウンロードしたcomment.xlsxをドラッグ &ドロップしてアップロードする
    - ドラッグ &ドロップができない場合はアップロードボタンをクリックしてファイルを選択します
  - アップロード後は、再びフォルダマークをクリックしてファイルツリーは閉じてもよい



データの取り込み

- pandasではExcelファイルを直接読み込むことができます。
  - 新しくコードセルを追加して下記を記述します。

```
#pandasのインポート
import pandas as pd

file = pd.ExcelFile("comment.xlsx")
data = file.parse("回答")
data.head()
```

取り込んだデータには自動的に連番のindexが割り当てられます。

	居住地	自由回答
0	愛知県	お勧めの時間帯は早朝です。富士山が世界遺産に指定されて以降、昼間の時間帯は大渋滞、駐車場が空...
1	愛知県	晴れていて絶景を一望できました。また足元は舗装されているので靴じゃなくても楽しめます。ヒール...
2	愛知県	水の流れの清らかさマイナスイオンを感じる穏やかな空気の流れ。駐車場からも適当な距離である。階...
3	愛知県	工事前、工事後にそれぞれ期間を空けて行ってみた。別に早い時間に行ったわけでもないが偶然人は少...

自由回答から単語(名詞)を取り出す(1)

- 新しくコードセルを追加して下記を記述します。

```
comments = data["自由回答"].tolist()  #自由回答の列をtolist()でリスト型に変換
words = []                            #単語リスト(最初は空)
for c in comments:                    #コメントを1つずつcに取り出して繰り返し
    node = tagger.parseToNode(c)      #形態素解析を実行(最初の形態素がnodeに入る)
```

字下げ

- 解説
  - リスト型のcomments変数には [“お勧めの時間帯は早朝です。・・・”,”晴れていて絶景を一望できました。・・・”, ・・・] などと自由回答がリスト形式で格納されます。
  - for c in comments: のfor文で、comments変数から1件ずつ自由回答が取り出されて変数cに代入されます。
  - 取り出した自由回答をtagger.parseToNodeで形態素解析して、最初の形態素をnode変数に代入しています。

自由回答から単語(名詞)を取り出す(2)

- さらに赤枠のコードを追記します。

```
comments = data["自由回答"].tolist()  #自由回答の列をtolist()でリスト型に変換
words = []                            #単語リスト(最初は空)
for c in comments:                    #コメントを1つずつcに取り出して繰り返し
    node = tagger.parseToNode(c)      #形態素解析を実行(最初の形態素がnodeに入る)
    while node:                       #形態素がなくなるまで繰り返し
        hinshi = node.feature.split(",")[0]  #品詞を抽出
        if hinshi in ["名詞"]:         #品詞が名詞ならば
            words.append(node.surface)  #wordsリストに表層形を追加
        node = node.next              #次のnodeを新たなnodeとする
print(len(words))                     #単語の件数
print(words[:20])                     #最初の20件の単語を表示
```

字下げ

- 解説

feature 名詞,普通名詞,副詞可能,*,*,*,キョウ,
0番目

  - 各形態素のfeatureを.split(“,”)でカンマで区切ってリスト化し、その0番目を[0]として取り出して品詞をhinshiに代入しています。
  - if文で、hinshiが「in リスト」のリスト項目に含まれていれば、wordsリストに追加しています。結果として品詞が名詞の場合にwordsリストに追加されます。

結果
```
5973
['時間', '早朝', '富士', '山', '世界', '遺産', '指定', '以降', '昼間', '時間', '渋滞',
```


単語(名詞)の件数をカウントする

- 形態素解析した結果から、単語の件数を数えましょう。
  - 件数を数える際には、Pythonの標準モジュールであるcollectionsのCounter関数が使用できます。
  - 新しくコードセルを追加して下記を記述します。

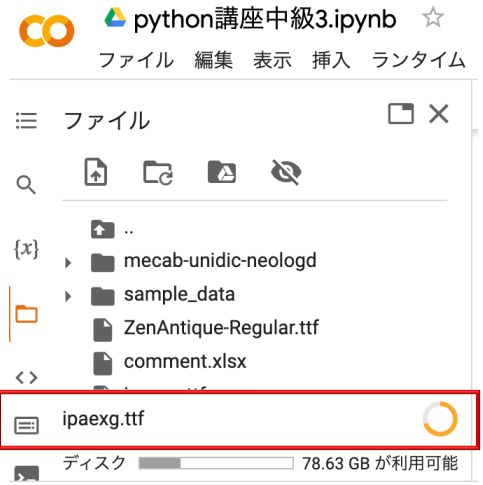
```
from collections import Counter #Counter関数のインポート
hindo = Counter(words)          #単語の出現頻度の集計
print(hindo)
```

```
Counter({'滝': 524, '駐車': 213, '白糸': 143, 'こと': 104, '富士': 99, '近く': 83, '山': 81, '円': 75,
```

「滝」という単語が524件抽出されたことを表しています。

ワードクラウドを描く(1)

- 出現頻度の多い単語ほど、大きな文字サイズで描いた図が「ワードクラウド」です。
- Pythonでは、wordcloudモジュールのWordCloudクラスでワードクラウドが描けます。
- 日本語でワードクラウドを描くには、別途、日本語フォントが必要です。
  - 前回、matplotlibで描くグラフを日本語対応させた方法とは別の手順が必要となります。
  - 手順:
    - 資料電子データサイトから、2つのフォントファイル「ipaexg.ttf」「ZenAntique-Regular.ttf」をダウンロードします。
    - ダウンロードした2つのフォントファイル「ipaexg.ttf」「ZenAntique-Regular.ttf」を、Google Colabのフォルダツリーにドラッグドロップしてアップロードします。



(補足)フォントファイルの入手先

- ipaexg.ttf
  - 文字情報技術促進協議会が公開しているフリーのフォント
  - https://moji.or.jp/ipafont/ipaex00103/

あのイーハトーヴォの
すきとおった風、
夏でも底に冷たさをもつ青いそら、
うつくしい森で飾られたモリーオ市、
郊外のぎらぎらひかる草の波。

- ZenAntique-Regular.ttf
  - Googleが公開しているフリーのフォント
  - https://fonts.google.com/specimen/Zen+Antique

あのイーハトーヴォの
すきとおった風、
夏でも底に冷たさをもつ青いそら、
うつくしい森で飾られたモリーオ市、
郊外のぎらぎらひかる草の波。

ワードクラウドを描く(2)

- 新しくコードセルを追加して下記を記述します。

```
import matplotlib.pyplot as plt
from wordcloud import WordCloud  #WordCloudクラスのインポート

#フォントファイルの指定(どちらか2種類切り替え)
fpath = "ipaexg.ttf"
#fpath = "ZenAntique-Regular.ttf"

#WordCouldオブジェクトの生成
wordcloud = WordCloud(background_color="white",  #背景色
                      width=600,                 #横幅
                      height=500,                #高さ
                      max_font_size=150,         #フォントの最大サイズ
                      font_path=fpath,           #フォントファイル
                      colormap="plasma")         #色合い
wordcloud.generate_from_frequencies(hindo)       #hindoを元にワードクラウドを生成

plt.figure(figsize=(15,12))     #wordcloud画像の出力サイズ指定(幅と高さの実寸をインチ指定)
plt.imshow(wordcloud)           #wordcloud画像を出力
plt.axis("off")                 #縦軸と横軸は非表示
plt.show()                      #出力の確定
```


出力例

ipaexg.ttf

ZenAntique-Regular.ttf

- colormap="plasma"の"plasma"を変えると、色合いが変わります。
  - viridis
  - inferno
  - magma
  - spring
  - summer
  - ・・・
- 引数に、prefer_horizontal=1 を加えると横書き固定となります。

付録

（付録）ストップワードの設定（1）

- 演習で描いたワードクラウドを確認すると、「0」や「こと」など、結果に含めたくない取るに足らない単語が見られます。
- このような除外したい単語のことを、ストップワードと呼びます。
- ストップワードを除外するには、あらかじめストップワードの一覧ファイルを用意します。
  - 例：資料電子データサイトに、日本語での一般的なストップワードの一覧ファイル「stopword.txt」を用意しました。

1	あそこ	11	いま	21	かやの
2	あたり	12	いや	22	から
3	あちら	13	いろいろ	23	がら
4	あっち	14	うち	24	きた
5	あと	15	おおまか	25	くせ
6	あな	16	おまえ	26	ここ
7	あなた	17	おれ	27	こっち
8	あれ	18	がい	28	こと
9	いくつ	19	かく	29	ごと
10	いつ	20	かたち	30	こちら

…

(付録)ストップワードの設定(2)

- ストップワードのリスト「stopword.txt」をGoogle Colabのファイルツリーにドラッグ &ドロップまたはメニューからアップロードします。

- 20枚目のスライドのコードについて赤枠のコードを追加し修正します。

```
swords = []                                  #ストップワードの一覧リスト(最初は空)
f = open("stopword.txt")                     #ストップワードの一覧ファイルを開く
txt = f.readlines()                          #ストップワードの一覧ファイルを行毎に読みtxtリストに格納
f.close()                                    #ストップワードの一覧ファイルを閉じる
print(txt)                                   #確認用
swords = [line.strip() for line in txt]      #ストップワードの行末の改行文字「¥n」をstripメソッドで取る
print(swords)                                #確認用

comments = data["自由回答"].tolist()          #自由回答の列をtolist()でリスト型に変換
words = []                                   #単語リスト(最初は空)
for c in comments:                           #コメントを1つずつcに取り出して繰り返し
    node = tagger.parseToNode(c)             #形態素解析を実行(最初の形態素がnodeに入る)
    while node:                                  #形態素がなくなるまで繰り返し
        hinshi = node.feature.split(",")[0]      #品詞を抽出
        #品詞が名詞かつスットプワードで無いならば
        if hinshi in ["名詞"] and not node.surface in swords:
            words.append(node.surface)           #wordsリストに表層形を追加
```


(付録)ストップワードの設定の結果

- ストップワードにより取るに足らない単語を除外したならば、単語の件数を再集計しワードクラウドを再描画してみましょう。
- 手順:
  - 22枚目のスライドのコードを再実行します。
  - 25枚目のスライドのコードを再実行します。

（付録）課題

- 課題１：抽出する形態素について、品詞が「名詞」だけでなく「形容詞」と「感動詞」も含めてください。

- 課題２：分析対象のサンプルを「静岡県」居住者に絞ってください。

（付録）課題１回答例

- 29枚目のスライドに対して下記赤枠のコードを追記して実行します。
- 25枚目のスライドのコードを再実行します。

```
swords = []                                  #ストップワードの一覧リスト（最初は空）
f = open("stopword.txt")                     #ストップワードの一覧ファイルを開く
txt = f.readlines()                          #ストップワードの一覧ファイルを行毎に読みtxtリストに格納
f.close()                                    #ストップワードの一覧ファイルを閉じる
print(txt)                                   #確認用
swords = [line.strip() for line in txt]      #ストップワードの行末の改行文字「¥n」をstripメソッドで取る
print(swords)                                #確認用

comments = data["自由回答"].tolist()          #自由回答の列をtolist()でリスト型に変換
words = []                                   #単語リスト（最初は空）
for c in comments:                           #コメントを1つずつcに取り出して繰り返し
    node = tagger.parseToNode(c)             #形態素解析を実行（最初の形態素がnodeに入る）
    while node:                                    #形態素がなくなるまで繰り返し
        hinshi = node.feature.split(",")[0]        #品詞を抽出
        #品詞が名詞かつスットプワードで無いならば
        if hinshi in ["名詞", "形容詞", "感動詞"] and not node.surface in swords:
```

（付録）課題2回答例

- 講座1日目で説明したように、pandasデータではフィルタをかけることができます。
- 19枚目のスライドに対して下記赤枠のコードを追記して実行します。

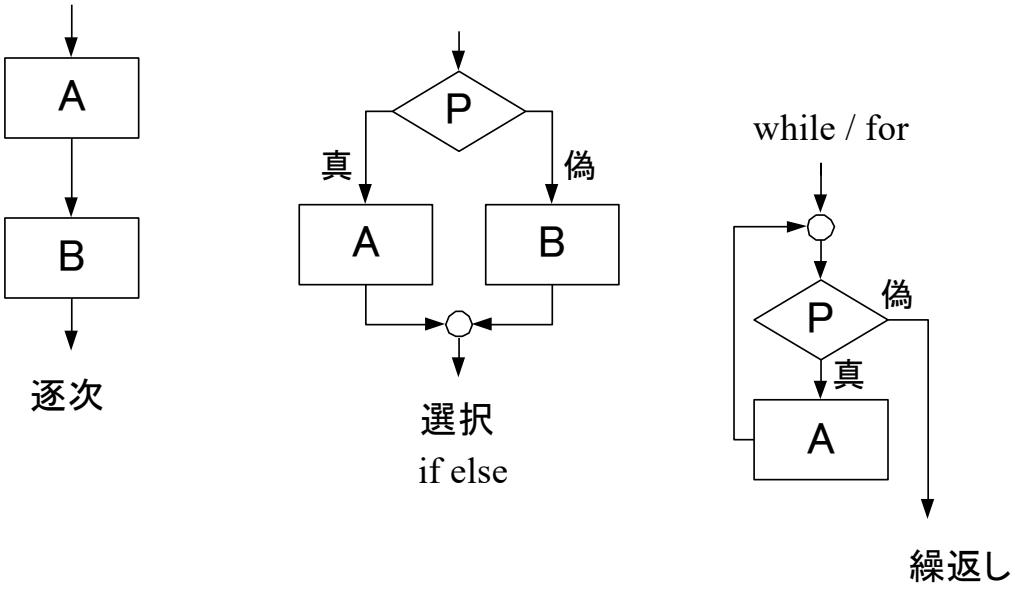
```
#pandasのインポート
import pandas as pd

file = pd.ExcelFile("comment.xlsx")
data = file.parse("回答")
data.head()

data = data[data["居住地"] == "静岡県"]
data
```

- 20枚目、21枚目、22枚目、25枚目のスライドのコードを再実行します。

（初級レベルの制御構文の確認）

（初級レベル）制御構文：逐次、選択、繰り返し

- Pythonでは、3つの制御構文を組み合わせてプログラムの流れ（フロー）を記述します。



（初級レベル）選択（if文とelse文）

- if文により、条件によって実行する処理を切り替えることができます。
  - 構文

```
if 条件式 :
    条件式が成り立つ場合に実行する処理1
    条件式が成り立つ場合に実行する処理2
    ・・・
else :
    条件式が成り立たない場合に実行する処理1
    条件式が成り立たない場合に実行する処理2
    ・・・
```

成り立つ場合に実行されるブロック

成り立たない場合に実行されるブロック

ブロックは行頭を等しく字下げします（インデント）

  - 条件によって実行したい処理の塊を「ブロック」と呼びます。
  - ブロックはカッコなどで囲んで範囲を表すのではなく、行頭を等しく字下げすることで範囲を表します。
  - 成り立たない場合の処理はelse文のブロックに書きます。成り立たない場合の処理がなければelse文とそのブロックは不要です。

（初級レベル）条件式の書き方

- 比較演算子
  - 2つの変数・定数同士の大小関係を比較する演算子です。

比較演算子	読み方	使い方
==	等しい	a == b
<	小なり	a < b
>	大なり	a > b
<=	以下	a <= b
>=	以上	a >= b
!=	等しくない	a != b

```
tokuten = 90
if tokuten >= 80:
  print("合格")
else:
  print("不合格")
```

```
合格
```

- in演算子
  - inの後ろのリストに値が含まれていたら成り立つ条件式が記述できます。

```
pref = "神奈川県"
if pref in ["神奈川県", "東京都", "千葉県", "埼玉県"]:
  print("首都圏です")
else:
  print("首都圏ではありません")
```

```
首都圏です
```


（初級レベル）繰り返し（while文）

- while文により、繰り返し処理（ループ処理）を記述することができます。
  - ある条件式が成り立っている間だけ、ブロックに書かれた処理を繰り返し実行します。
  - 構文

成り立つ場合に繰り返し実行されるブロック

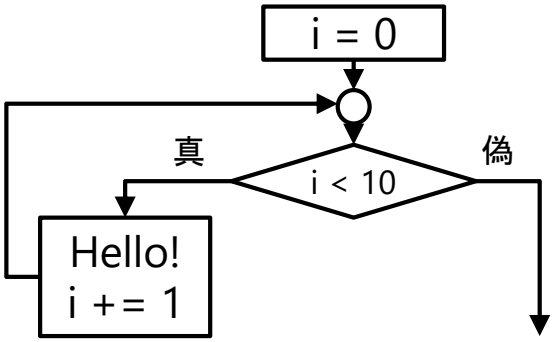
```
while 条件式 :
    条件式が成り立つ場合に実行する処理1
    条件式が成り立つ場合に実行する処理2
    ・・・
```

ブロックは行頭を等しく字下げします（インデント）

  - 例：Helloを10回表示する

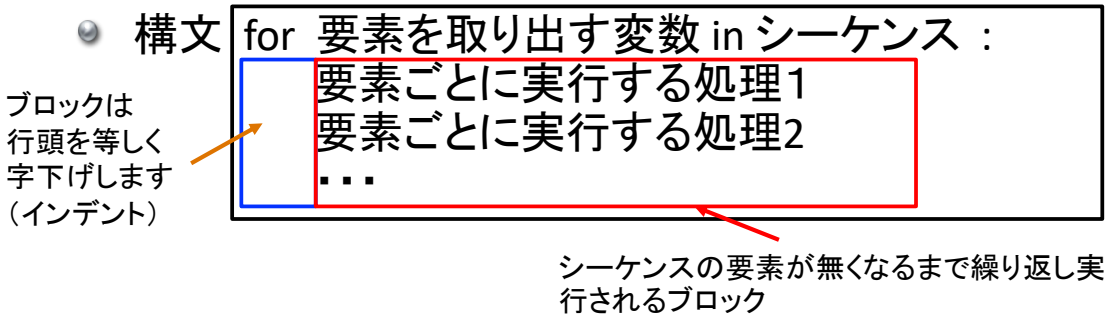
```
i = 0
while i < 10:
  print("Hello!" + str(i))
  i += 1
```

```
Hello!0
Hello!1
・・・
```



（初級レベル）繰り返し（for文）

- for文では、シーケンスから要素を1つずつ取り出して、要素がなくなるまで処理を実行します。
  - シーケンスとは？
    - 複数の要素を持ちインデックスで要素の指定ができるデータ型の総称で、文字列、リスト、タプルが該当します。
  - 構文

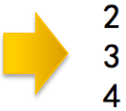


  - 例：

```
for i in [1, 2, 3, 4, 5]:
    print(i)
```

結果はどちらも同じ

```
for i in (1, 2, 3, 4, 5):
    print(i)
```



```
1
2
3
4
5
```