

# コンピュータリテラシ発展 ～Pythonを学ぶ～

## ： 第9回：Excel作業の前工程・後工程の自動化

メディア

( [shimizu@info.shonan-it.ac.jp](mailto:shimizu@info.shonan-it.ac.jp) )

# 今回の授業内容

# 今回の授業内容

- 前 の
- 
- CSVデータの
-

## 前回の課題解説

## 前回の課題解説

- 前 の の を します
- について があればご ください

## 解答例

[https://colab.research.google.com/drive/1il56vahBZGQGQwkpblwzJ92\\_Eodw6xGf?  
usp=sharing](https://colab.research.google.com/drive/1il56vahBZGQGQwkpblwzJ92_Eodw6xGf?usp=sharing)

# 文字列操作

# 文字列操作

- Pythonで      の      を行います
  - から      定の      の      出や      が      にできるようになります
  - **正規表現**を      してより発      的な      などを      にします
- について      しくかつわかりやすく      しているサイト
  - <https://www.tohoho-web.com/ex/regex.html>

## 利用操作

の前に対 となるテキストファイルとそれを いておくフォルダを  
す

Driveの 業 フォルダに「**text\_search**」というフォルダを

**search**」フォルダに「**file.txt**」ファイルを

ファイル は の りです

東 タワーの は で、東 スカイツリーの は です。



# 文字列操作

- の前に対   となるテキストファイルとそれを   いておくフォルダを  
          します

業場   に「                   」フォルダを   する

「                   」フォルダに「                   」ファイルを   して   を入   する

東   タワーの           は                   で、東   スカイツリーの           は                   です。

# 文字列操作

- 文字列がふくまれるかどうかを 判定
- **in演算子**
  - 文字列の 文字列が 含まれるかどうかを 判定
  - 含まれる場合：**True**
  - 含まれない場合：**False**

# 文字列操作

- が 　　まれている 　　を 　　定
- `find()` メソッド
  - の 　　が 　　まれる 　　を 　　す
  - は「0」から 　　え 　　める

位置	0	1	2	3	4	...
文字	東	京	タ	ワ	ー	...

# 文字列操作

- を って「タワー」という の と を
- がふくまれるかどうかを 定 : **in**演算子
- の を 定 : `find()` メソッド

タワー

タワー  
タワーという タワー が に 含まれています

タワーという は 含まれていません

# 正規表現を使って文字列を検索

- とは
  - のパターンを する 方
  - サイト
    - <https://w.wiki/5SGg>
    - <https://docs.python.org/ja/3/library/re.html>
    - <https://www.tohoho-web.com/ex/regexp.html>

# 正規表現を使って文字列を検索

- とは
  - 「^ (キャレット)」 「. (ピリオド)」 「\* (アスタリスク)」 「\d」 などの 特殊な記号を使ったメタ文字と 文字列 (リテラル) の 組み合わせで 文字列を検索します
  - メタ文字はたくさんあります

# メタ文字の種類と意味

表現	意味
.	行をくすすべてののいずれか1
*	前のを0上りし(0=そのがない場合)
{N}	前のパターンのNの
^	行の
\$	行の
A B	AかBのいずれか1

# メタ文字の種類と意味


[x-y]	x に 定めた 1 つの文字と y に 定めた 1 つの文字のいずれか 1
[x-yz]	「-」の 間に 定めた 1 つの文字と「z」の 間に 定めた 1 つの文字のいずれか 1
[^x-y]	x に 定めた X のいずれか 1
\d	1 , [0-9]と同
\D	の の 1
\w	すべてのアルファベットとアンダースコアのいずれか 1




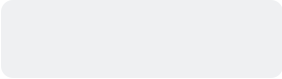
# 正規表現を使って文字列を検索

- の いかた
  - メタ を ってみましょう
  - は「**数字3桁-数字4桁**」で されています
  - メタ を うと「`\d\d\d-\d\d\d\d`」となります
  - によっては「**\ (バックスラッシュ)**」は「**¥ (円マーク)**」で されま  
す

# 正規表現を使って文字列を検索

- Pythonで           を   う
  -  モジュール
    - regular expressionの
    - インストール   み
    - インポート

# 正規表現を使って文字列を検索

- Pythonで            を    う
  -  モジュール
    - 
      - マッチした            をリストで    り出す
      - 1            :    出する
      - 2            :    対    となる

# 正規表現を使って文字列を検索

- Pythonで `re` を使う
  - `re` モジュール
    - `re.compile()`
    - **raw**
      - エスケープシーケンスをそのままの文字列として扱う
      - 「\」を扱う場合は, raw を使うのが一般的

# 正規表現を使って文字列を検索

- Pythonで `import` を う
  - モジュールの `__name__`

## モジュールをインポート


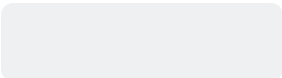
対 のファイルのパスを 定

ファイルを エンコーディングで み みモードで く

ファイルの を み り、 を って のリストを 出  
は の にマッチする

出された のリストを

# 正規表現を使って文字列を検索

- Pythonで            を    う
  -  モジュール
    - 
      - オブジェクトをつくれる
      - つくったオブジェクトは            も    える

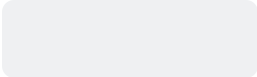
# 正規表現を使って文字列を検索

- Pythonで `import` を う
  - モジュールの

を す      パターンをオブジェクト  
             は                          の                          にマッチする

ファイルの を み り、 を って のリストを 出

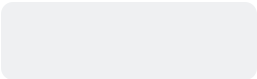
# 正規表現を使って文字列の位置を調べる

- 正規表現で文字列の位置を調べます
- 
  - 正規表現で文字列の位置を調べます
  - 1 : パターン
  - 2 : 対



# 正規表現を使って文字列の位置を調べる

- 正規表現を使って文字列の位置を調べる

- 

ファイルの位置を調べる、正規表現を使って文字列の位置を調べる

# 正規表現を使って文字列の位置を調べる

- `match()` で ー した の を める
- `match()` メソッド
  - マッチオブジェクト
    - `match()`: マッチした
    - `start()`:
    - `end()`:
    - `group()`: と をタプルで り出す

# 正規表現を使って文字列の位置を調べる

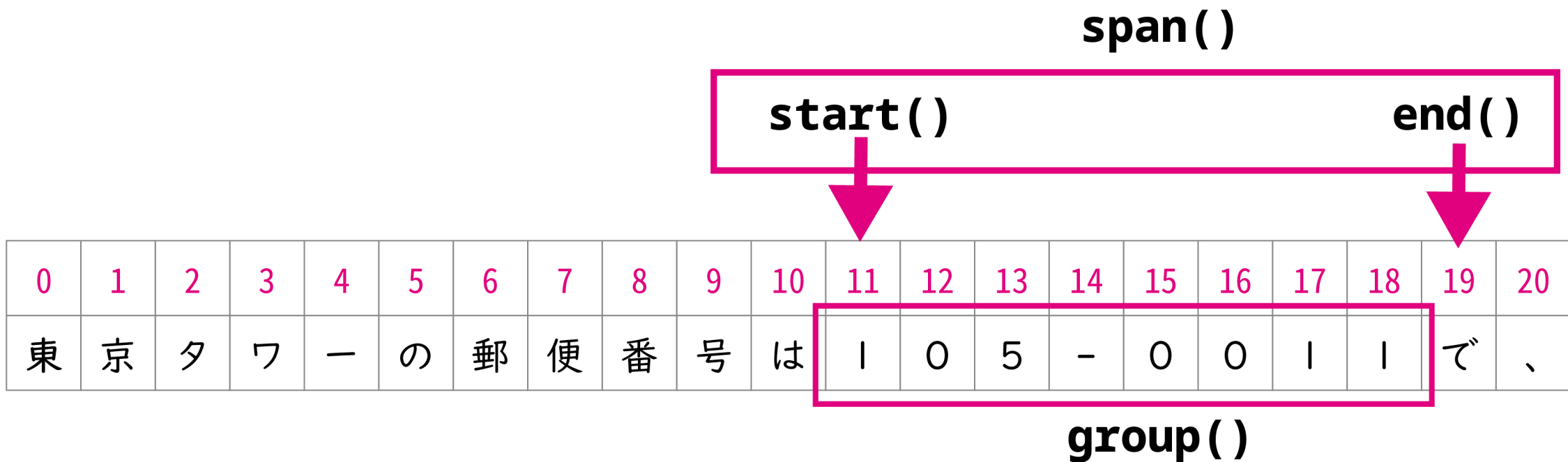
- `re.match()` で 一致した の を める
- `re.findall()` メソッドとマッチオブジェクト

ファイルの 内容を、正規表現を使って 文字列にマッチする かどうかを調べる

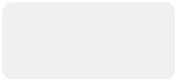
マッチする が見つかった場合  
マッチした の オブジェクトを  
マッチした 自己を返す  
マッチした の を返す  
マッチした の を返す  
マッチした の を返す  
のタプルを返す

# 正規表現を使って文字列の位置を調べる

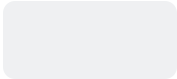
- `match()` で 一致した ものを 調べる
- `match()` メソッドとマッチオブジェクト



# 文字列の置換

- してヒットした を の に き えます
- 
  - 1 : パターン
  - 2 : 後
  - 3 : 対 のテキスト

# 文字列の置換

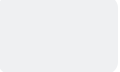
- してヒットした を の に き える
- 
- 「東京」を「**Tokyo**」に き える

ファイルの をすべて み り、 を って を  
東 という を  
東

後のテキストを

# CSVデータの処理

# CSVの出力

- CSVファイルとは
  - Comma-Separated Valuesの
  - り     であるカンマ「,」で     ったデータ     です
-  モジュール
  - pythonでCSVを     を行うにはcsvモジュールを     します
  - インストール   み, インポート



# CSVの出力

- 「**CSV**」フォルダを して**CSV**ファイルを します
- **CSV**ファイルにデータを き むには を します
- の にファイルオブジェクトと をわたします
- は, り のことでデフォルトでは「」を します
- メソッドはリストを として 定し, 行 でCSVに出 します

# CSVの出力

- 「**CSV**」フォルダを として「**sample.csv**」ファイルを します

モジュールをインポート

ファイルを するためのフォルダのパスを 定

定されたパスにフォルダを . に する場合はエラーを出さずに する

ファイルを モード で き, 行コードを 定してファイルオブジェクトを

オブジェクトを し, カンマ り に 定

行 のデータ を ファイルに き む

行 のデータ を ファイルに き む

# CSVの出力

- CSVファイルの読み込みには `csv` モジュールの `DictReader` オブジェクトを使用します
- CSVファイルの各行をリストのイテラブルとして扱うことができます

ファイルをメモリモードで開く

オブジェクトを

ファイルの各行を

行のリストとして

# CSVの加工

- CSVファイルの 読み込みと出力ができるようになったのでデータの 読み込みと出力ができます
- データの 読み込み方法は色々ありますがここでは後に 詳しく 説明する `csv` ライブラリを 使います
- `csv` ライブラリではCSVファイルの 読み込み・出力 もできます

# CSVの加工

- pandasを使ってCSVファイルをみる：`pandas.read_csv()`
- CSVをみる、`pandas.DataFrame`というオブジェクトを返します
- `pandas.DataFrame`はデータの列をもち、行0のデータをきえるときは「`index`」ときます
- `pandas.DataFrame.to_csv()`メソッドでCSVファイルにき出して保存します
- 「`index`」「`columns`」はみみのに自分でされるをにする定です

# CSVの加工

- pandasを使ってCSVを読み込み、データを整形して出力する

モジュールをインポート

ファイルを指定し、データフレームに  
は、ファイルにヘッダー行がないことを指定する

データフレームの特定の行（インデックス）を抽出してデータを取得できる

抽出されたデータフレームを指定したファイルに  
は、行のインデックスを指定しないことを指定する  
は、ヘッダー行を指定しないことを指定する

# CSVの加工

- pandasについては後の 業で します
- pandasはとても なるライブラリでとても くのことができます
- 業ではそのほんの一 のみを います
- しい を りたい 生は やQiitaなどをみてみてください
- <https://pandas.pydata.org/>
- <https://qiita.com/tags/pandas>

# 課題



# 課題

- Moodleにある「SCfCL-9th-prac.ipynb」ファイルをダウンロードしてColabにアップロードしてください
- が したら「File」>「Download」>「Download .ipynb」で「.ipynb」でダウンロードしてください
- ダウンロードした **.ipynbファイル** と した「Prac09」フォルダを して Moodleに 出してください
- 出 は **6月20日(木) 20時まで** です