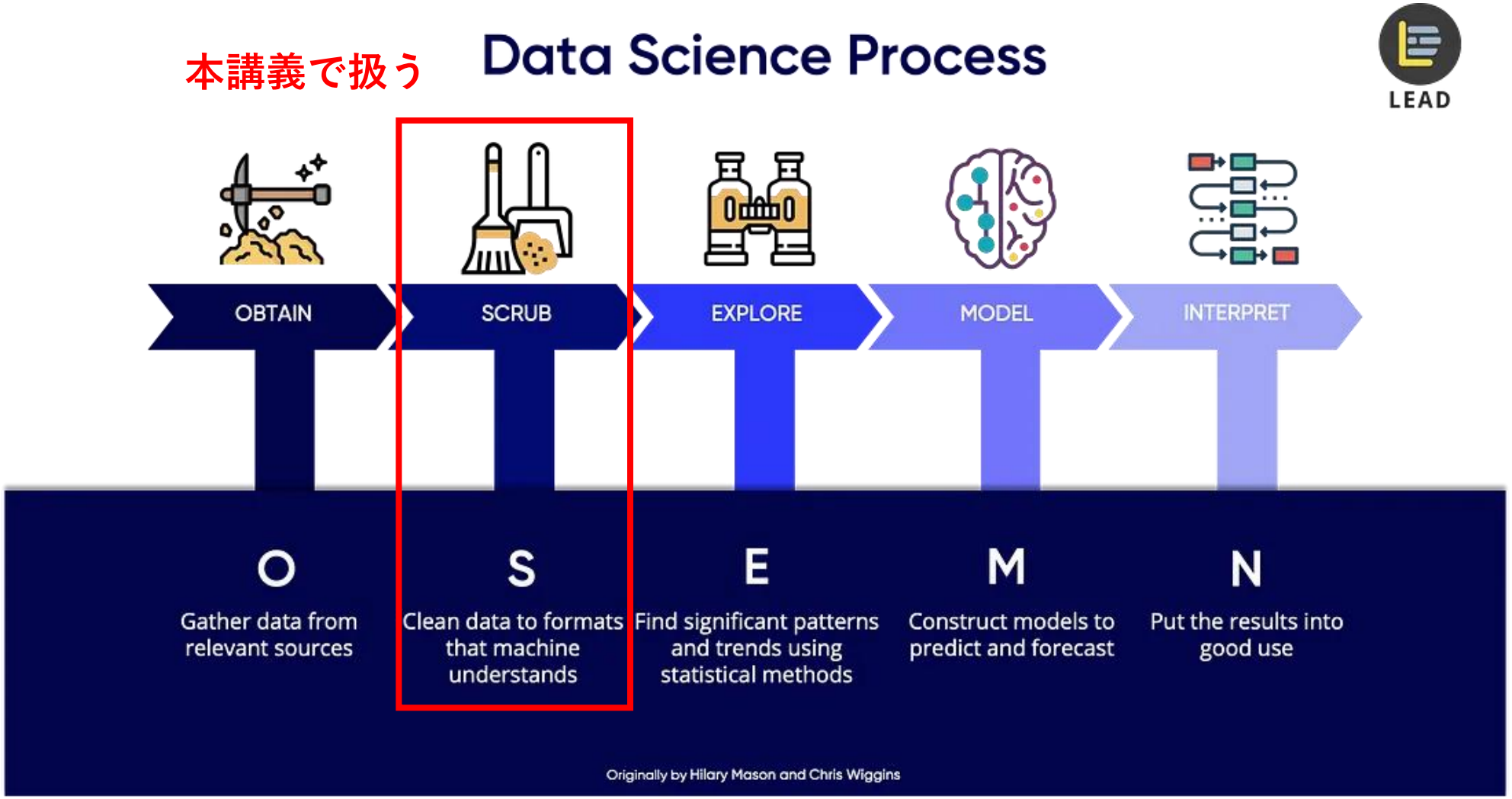
Week3 Pythonによるデータ加工処理の基礎（Pandas）

講師：鯛 涼太

データサイエンスとPandas

本講義で扱う

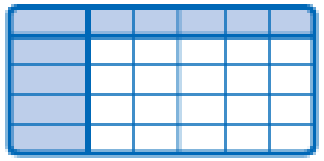


https://towardsdatascience.com/5-steps-of-a-data-science-project-lifecycle-26c50372b492

データの種類

本講義で扱う

構造化データ

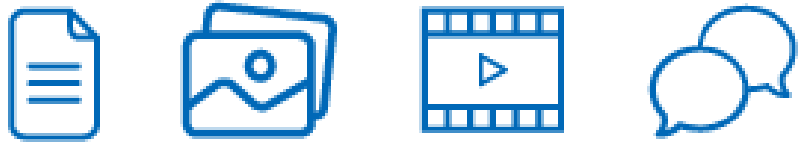


固定長ファイル　Excel / CSV

RDB内のデータ

二次元の表形式など
値が数値・記号で
1テーブルに整理されている

非構造化データ



テキスト　音声　画像　動画　センサーログ

半構造化データ

XML　JSON　html

表形式ではないが
データに規則性がある

データに規則性がなく
表形式に変換できない

https://solution.toppan.co.jp/digital/contents/cdp_contents07_0831.html

Pandasとは

- データを数表として扱うことを可能にするライブラリ
- **データフレーム形式**で様々なデータの加工が可能になる
- 簡単に言えば，Excelを操作するようにデータを扱える

Pandasの強み

- 数値データや時系列データ，文字列を処理する関数が揃っている
- データ集約などの統計処理に強い
- csv, excel, jsonなど様々なファイル形式を読み書きすることが可能

Pandasの弱み

- 計算速度が遅い
- メモリ消費が激しい

Pandasのデータ構造

■ Pandasでよくつかわれるデータ構造はSeriesとDataFrame

Series (1次元)

		ID
1行目	0	100
2行目	1	101
3行目	2	102
4行目	3	103
5行目	4	104

DataFrame (2次元)

		1列目 ID	2列目 City	3列目 Birth_year	4列目 Name
1行目	0	100	Tokyo	1990	Hiroshi
2行目	1	101	Osaka	1989	Akiko
3行目	2	102	Kyoto	1992	Yuki
4行目	3	103	Hokkaido	1997	Satoru
5行目	4	104	Tokyo	1982	Steve

Pandasのデータ構造

■ データ本体のvaluesに行のラベルindexや列のラベルcolumnsをつけた構造

Series (1次元)

		ID
1行目	0	100
2行目	1	101
3行目	2	102
4行目	3	103
5行目	4	104

(index) (values)

DataFrame (2次元)

		1列目 ID	2列目 City	3列目 Birth_year	4列目 Name
1行目	0	100	Tokyo	1990	Hiroshi
2行目	1	101	Osaka	1989	Akiko
3行目	2	102	Kyoto	1992	Yuki
4行目	3	103	Hokkaido	1997	Satoru
5行目	4	104	Tokyo	1982	Steve

(columns)

(index) (values)

演習

- PandasではDataFrame形式を扱うメソッドが数多くあります.

本講義で扱う内容・キーワード

データの選択と代入 / データの抽出 / 値のソート / データの結合
データの削除 / データの集約とグループ演算 / 階層型インデックス
欠損値の取り扱い / 時系列データの取り扱い

- もちろん全てを覚える必要はありません.
- Pandasを使ってどのようなことができるのか,
  大まかなイメージを理解することが本日の目標になります.

実際にコードを動かしてみましょう
教材のノートブックを開いてください

データの選択

	ラベル指定	インデックス指定
Seriesや DataFrame を取得する	**loc** df.loc[0:3, ['ID','Birth_year']]	**iloc** df.iloc[0:4, [0,2]]
単独の要素を取得する	**at** df.at[2,'Birth_year']	**iat** df.iat[2, 2]

loc df.loc[0:3, ['ID','Birth_year']]

	ID	City	Birth_year	Name	Score
0	100	Tokyo	1990	Hiroshi	0
1	101	Osaka	1989	Akiko	10
2	102	Kyoto	1992	Yuki	20
3	103	Hokkaido	1997	Satoru	30
4	104	Tokyo	1982	Steve	40

iloc df.iloc[0:4, [0,2]]

	ID	City	Birth_year	Name	Score
0	100	Tokyo	1990	Hiroshi	0
1	101	Osaka	1989	Akiko	10
2	102	Kyoto	1992	Yuki	20
3	103	Hokkaido	1997	Satoru	30
4	104	Tokyo	1982	Steve	40

at df.at[2,'Birth_year']

	ID	City	Birth_year	Name	Score
0	100	Tokyo	1990	Hiroshi	0
1	101	Osaka	1989	Akiko	10
2	102	Kyoto	1992	Yuki	20
3	103	Hokkaido	1997	Satoru	30
4	104	Tokyo	1982	Steve	40

iat df.iat[2, 2]

	ID	City	Birth_year	Name	Score
0	100	Tokyo	1990	Hiroshi	0
1	101	Osaka	1989	Akiko	10
2	102	Kyoto	1992	Yuki	20
3	103	Hokkaido	1997	Satoru	30
4	104	Tokyo	1982	Steve	40

階層型インデックス

	店名	場所	商品名	色	売上
0	A店	大阪	商品1	赤	200.0
1	A店	大阪	商品1	青	0.0
2	A店	大阪	商品2	赤	100.0
3	A店	東京	商品1	赤	500.0
4	A店	東京	商品1	青	300.0
5	A店	東京	商品2	赤	400.0
6	B点	大阪	商品1	赤	800.0
7	B点	大阪	商品1	青	600.0
8	B点	大阪	商品2	赤	700.0

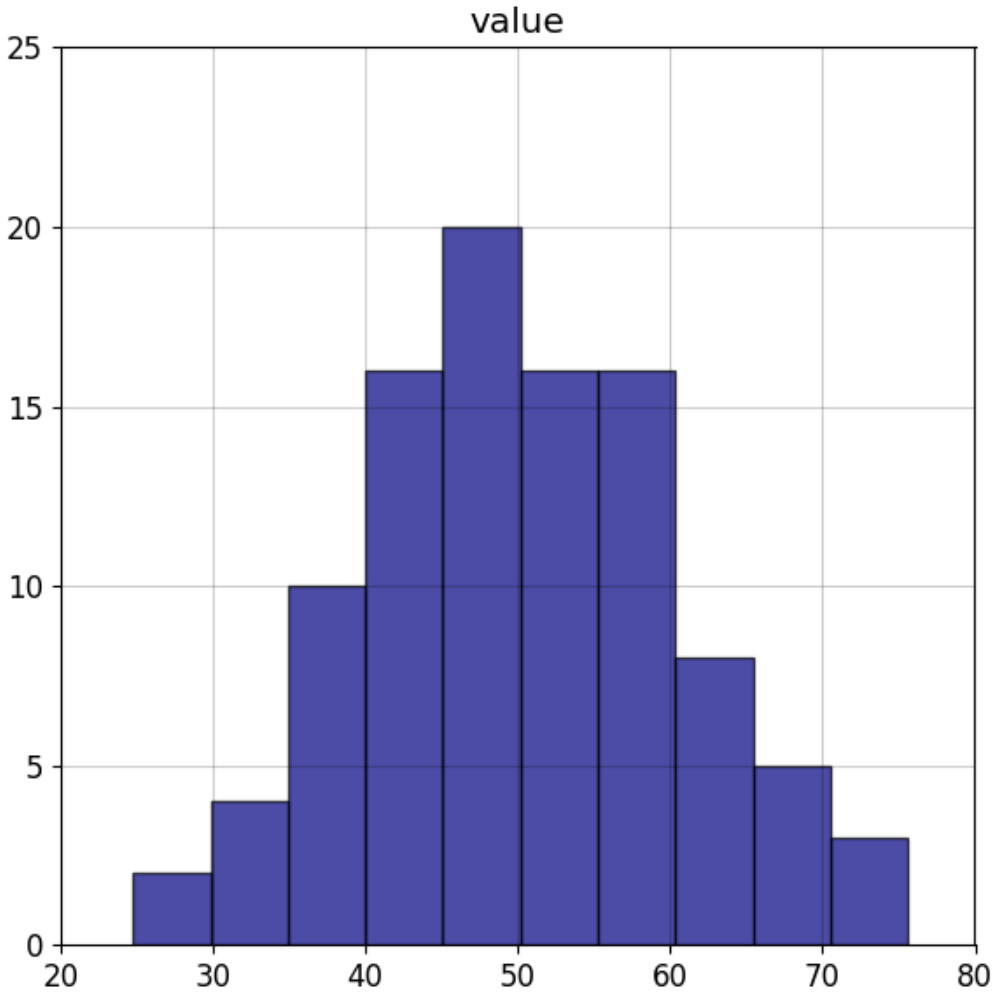
		商品名	商品1	商品2	商品1	level 0
		色	青	赤	赤	level 1 (level -1)
店名	場所					
A店	大阪		0	100	200	
	東京		300	400	500	
B点	大阪		600	700	800	

level 0 level 1 (level -1)

- データ自体を変えずに，視認性が向上する
- フィルタリングや集計がしやすくなる

ビン分割

data = DataFrame({“value”:np.random.normal(50,10,100)}) # 正規分布に従う乱数



cut():データを等間隔で分割

pd.cut(data[“value”], bins=5)

	(24.702, 34.927]	(34.927, 45.101]	(45.101, 55.275]	(55.275, 65.449]	(65.449, 75.624]
count	6	26	36	24	8

qcut():データを等個数で分割

pd.qcut(data[“value”], q=5)

	(24.752, 42.199]	(42.199, 46.881]	(46.881, 53.138]	(53.138, 58.002]	(58.002, 75.624]
count	20	20	20	20	20