やまぐち
高校生データサイエンティスト育成講座

はじめに

みなさん
講座へようこそ

本講座は
AI・データサイエンス
を学ぶ！

本講座を終えると

1. AI/データサイエンスの入り口に立てる

本講座を終えると

1. AI/データサイエンスの入り口に立てる

2. データ分析プログラミングが経験できる

本講座を終えると

1. AI/データサイエンスの入り口に立てる

2. データ分析プログラミングが経験できる

3. ちょっとカッコよくなれる

たとえば

RootMeanSquaredError

Jupyter

Cross-Validation

Scikit-learn

Pandas

Python

DeepLearning

lightGBM

Seaborn

Xgboost

LinearRegression

たとえば

RootMeanSquaredError

Jupyter

Cross-Validation

Scikit-learn

Pandas

Python

DeepLearning

だいたいわかるようになります！

lightGBM

Seaborn

oost

LinearRegression

これから話すこと

① なぜAI・データサイエンスなのか？

これから話すこと

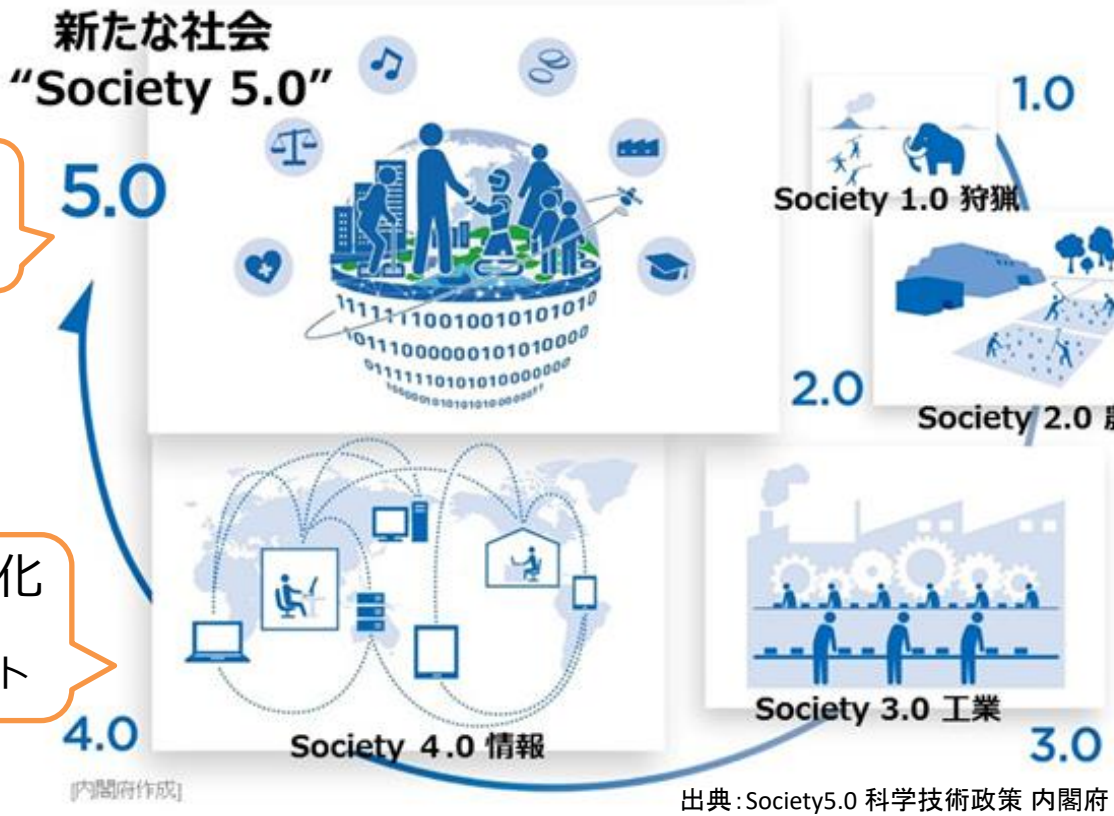
① なぜAI・データサイエンスなのか？

② 本講座で心がけていただきたいこと

なぜAI・データサイエンスなのか？

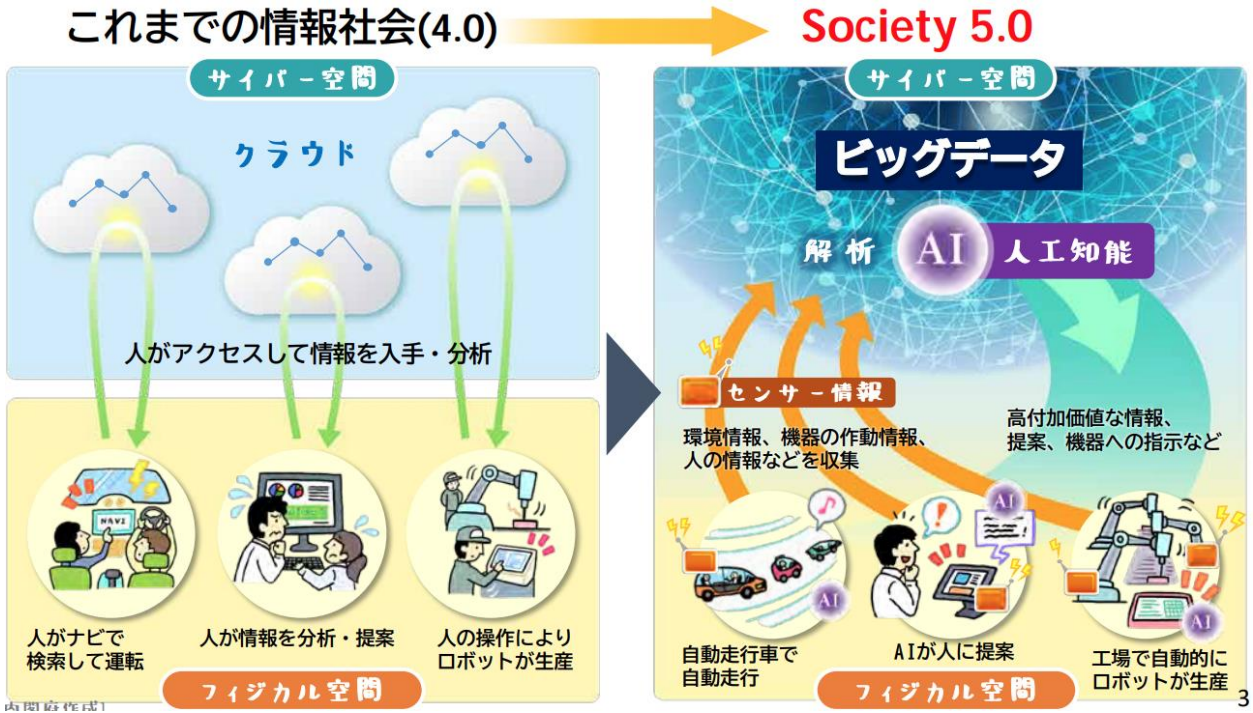
Society 5.0

Society X = 社会の変化を表したもの



出典：Society5.0 科学技術政策 内閣府

Society 5.0



出典：Society5.0 科学技術政策 内閣府

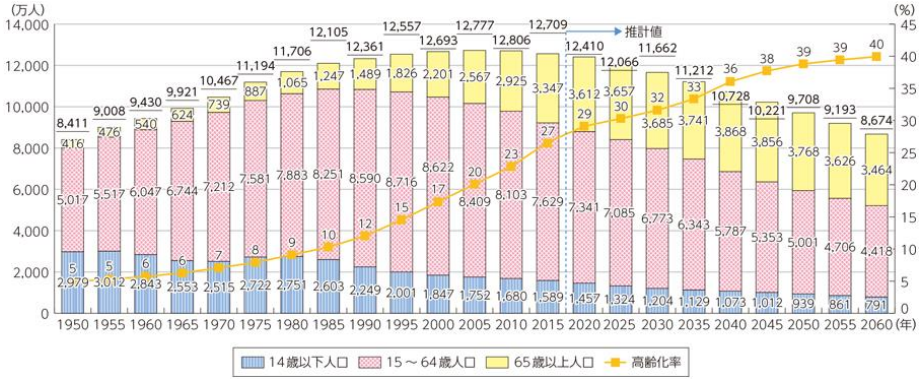
IoTや**5G**により
知識や情報が密に共有

ビックデータを通じて
必要な時に必要なものが
AIや**ロボット**、**ドローン**等により
自動的に提供される社会

なぜSociety5.0は必要？

▼少子高齢化

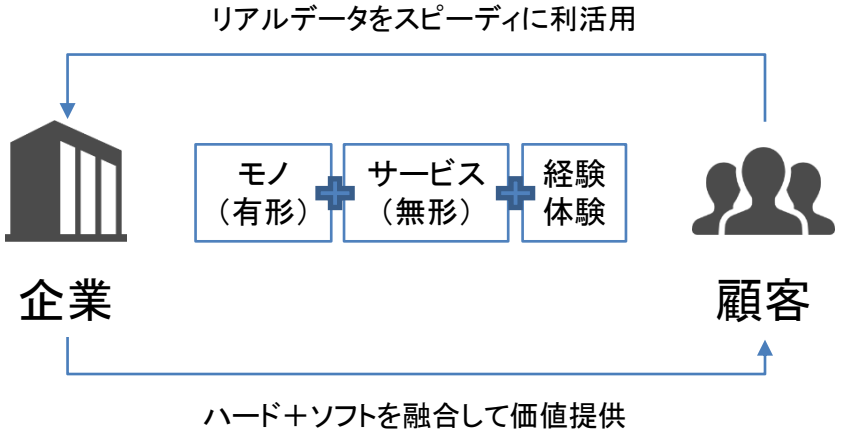
どんどん働き手が減っている！
しかし社会インフラ維持の為の労働力は減らない。むしろ増大。



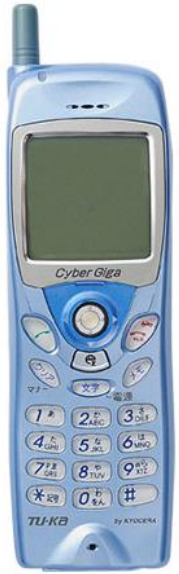
（出典）2015年までは総務省「国勢調査」（年齢不詳人口を含む）、2020年以降は国立社会保障・人口問題研究所「日本の将来推計人口（平成24年1月推計）」（出生中位・死亡中位推計）

▼ものづくり＋付加価値

データ利活用が進んでいる企業が経済に大きな影響を与えており、今後、より競争力が増していく。



高田の携帯電話の歴史



2000 2001 2002 2005 2007 2009 2011

モノクロ	画面カラー＋デジカメ	６４和音着うた	カメラ良くなる着うたフル	画面大きくなるカメラ高機能	スマホになる３G→4G、容量大

高田の携帯電話の歴史

機能は確実に進化し続けているが
段々と違いが分からなくなってきている
↓
機能だけでは競争に勝てない
「付加価値の提供」が必須の時代に。

2000	2001	2002	2005	2007	2009	2011
モノクロ	画面カラー ＋デジカメ	６４和音 着うた	カメラ良くなる 着うたフル	画面大きくなる カメラ高機能		スマホになる ３G→4G、容量大

高田の携帯電話の歴史

機能は確実に進化し続けているが
段々と違いが分からなくなってきている

AI・データ分析のスキル
が確実に武器になる時代

機能だけでは勝てない
「付加価値の提供」が必須の時代に。

2000	2001	2002	2005	2007	2009	2011
モノクロ	画面カラー ＋デジカメ	６４和音 着うた	カメラ良くなる 着うたフル	画面大きくなる カメラ高機能		スマホになる ３G→4G、容量大

②本講座で心がけて頂きたいこと

これから勉強することには、、、

これから勉強することには、、、

基本

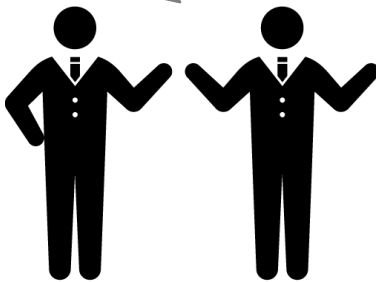
正解はありません

たとえば…

たとえば…

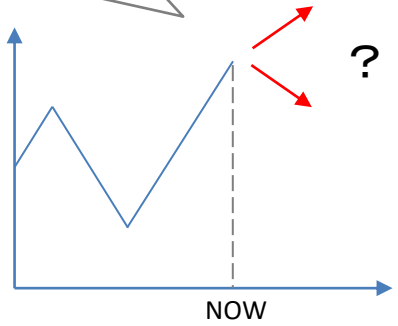
プログラミングの書き方

こっちの方が見やすい！
こっちが動作速くない？



データの解釈

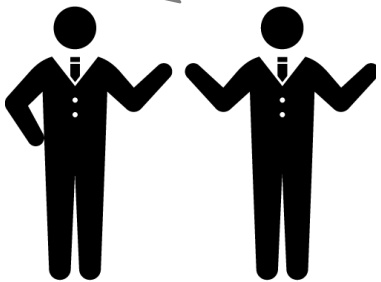
周期的に次は下がる！
いやそのまま上昇する！



たとえば…

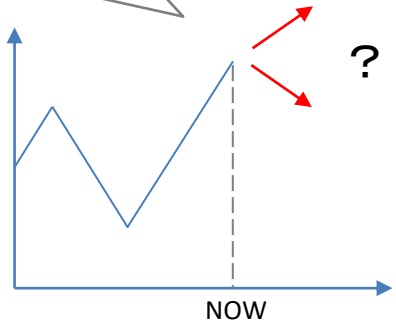
プログラミングの書き方

こっちの方が見やすい！
こっちが動作速くない？



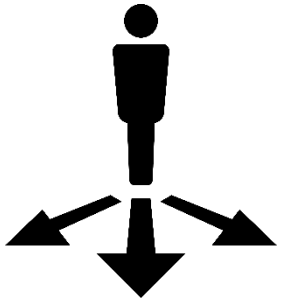
データの解釈

周期的に次は下がる！
いやそのまま上昇する！

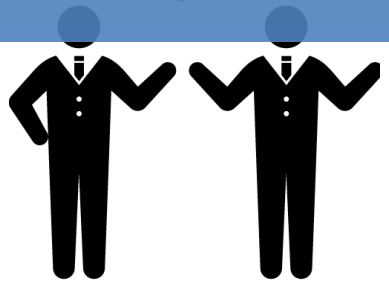
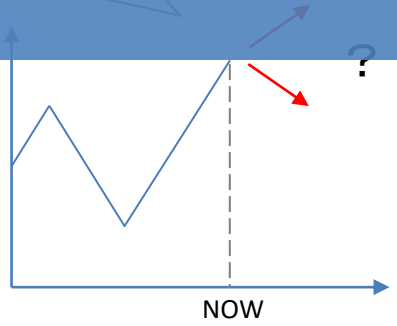
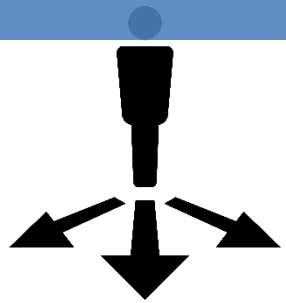


アプローチ方法

右が正解？左が正解？
いや真ん中もありかも･･･



たとえば…

プログラミングの書き方	データの解釈	アプローチ方法
こっちの方が見やすい！ こっちが動作速くない？	周期的に次は下がる！ いやそうまでは言えない！	右が正解？左が正解？ いや真ん中もありかも・・・
		

じゃあどうすればいいの？

心構え３か条

1. 試行錯誤をする・調べる

- やってみないとわからない
- 調査する力（ググる力）は本当に重要

心構え３か条

1. 試行錯誤をする・調べる
   - やってみないとわからない
   - 調査する力（ググる力）は本当に重要

2. 自分なりの意見／仮説を持つ・議論する
   - 考えの整理・洗練化

心構え３か条

1. 試行錯誤をする・調べる

- やってみないとわからない
- 調査する力（ググる力）は本当に重要

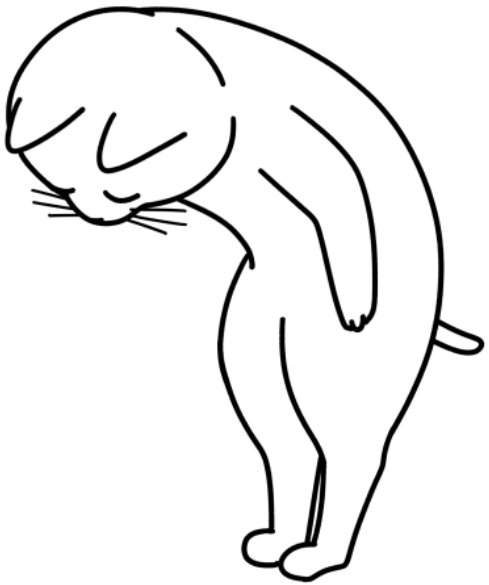
2. 自分なりの意見／仮説を持つ・議論する

- 考えの整理・洗練化

3. 質問する・教え合う

- “わからない”をそのままにしない
- 教えることができるようになる近道

もっと聞きたいこと・知りたいことがあれば slackで質問をお待ちしています！



さあデータサイエンスを始めましょう

本日のスケジュール

時間	内容
11：10 ～ 11：40	オリエンテーション
11：40 ～ 12：10	データサイエンティストは何者？
12：10 ～ 13：10	昼休憩
13：10 ～ 14：10	Slackに触ってみよう
14：10 ～ 14：20	休憩
14：20 ～ 15：20	Pythonに触ってみよう①
15：20 ～ 15：30	休憩
15：30 ～ 16：50	Pythonに触ってみよう②
16：50 ～ 17：00	閉講・アンケート

データサイエンティストは何者？

本講座の目的

将来を担う高校生の皆さんに対して特に、

データサイエンティスト
データサイエンスを使える人
・データサイエンスに係る事業ができる起業家

を目指してもらいたい、応援したい！

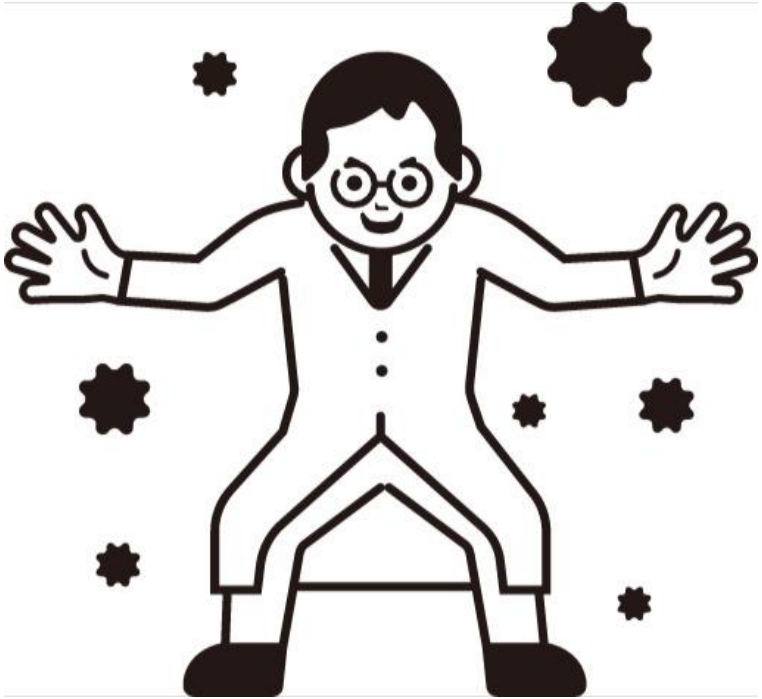
本講座の目的

将来を担う高校生の皆さんに対して特に、

データサイエンティスト
データサイエンスを使える人
・データサイエンスに係る事業ができる起業家

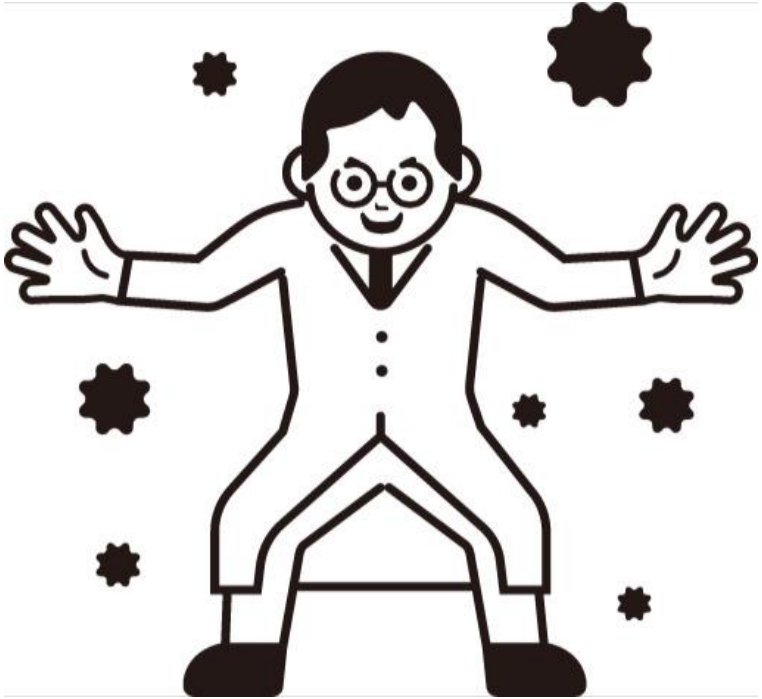
を目指してもらいたい、応援したい！

データサイエンティストってどんなイメージ？



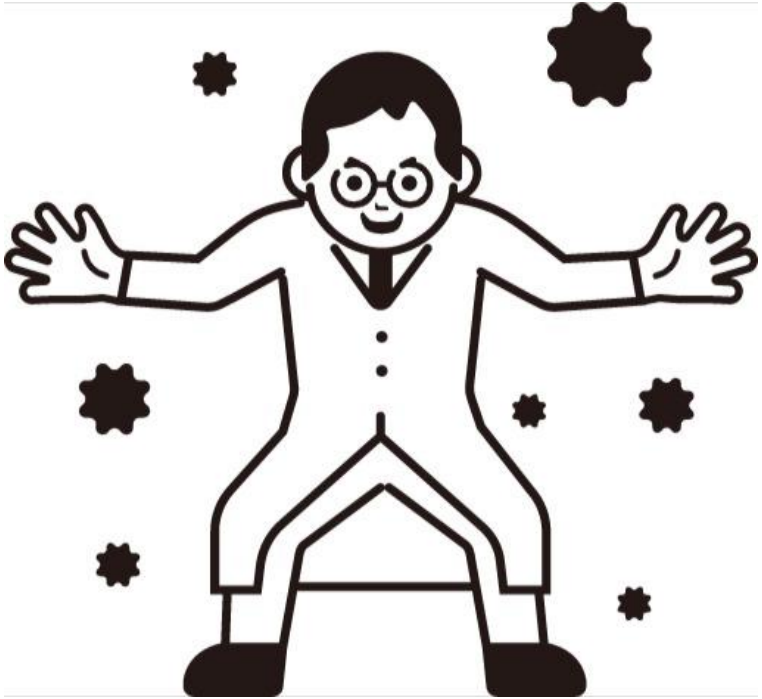
データサイエンティストってどんなイメージ？

数学・統計



データサイエンティストってどんなイメージ？

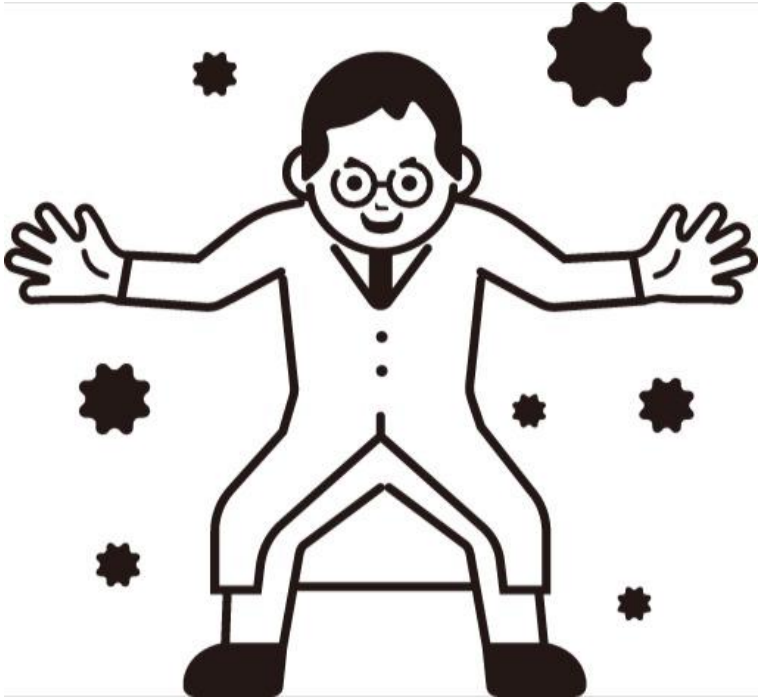
数学・統計



プログラム

データサイエンティストってどんなイメージ？

数学・統計

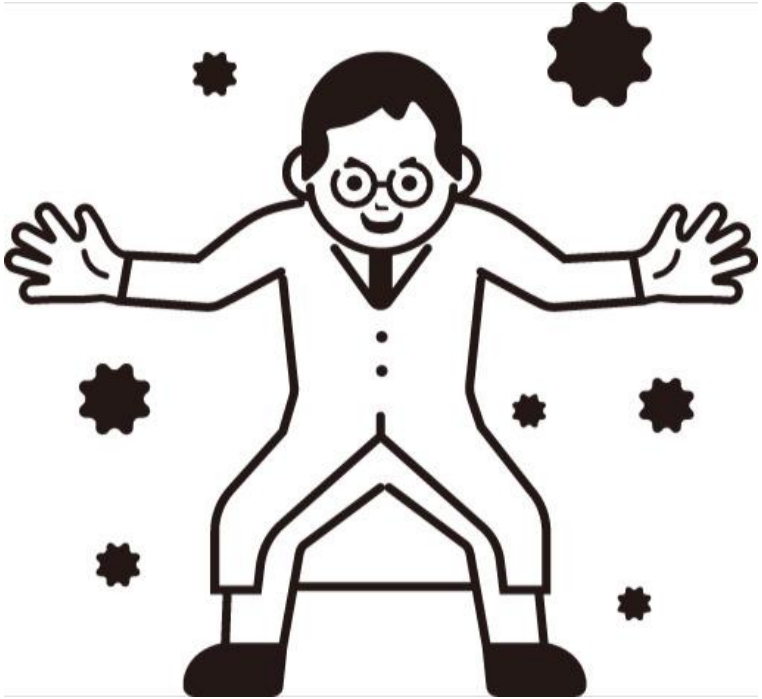


データ分析

プログラム

データサイエンティストってどんなイメージ？

数学・統計



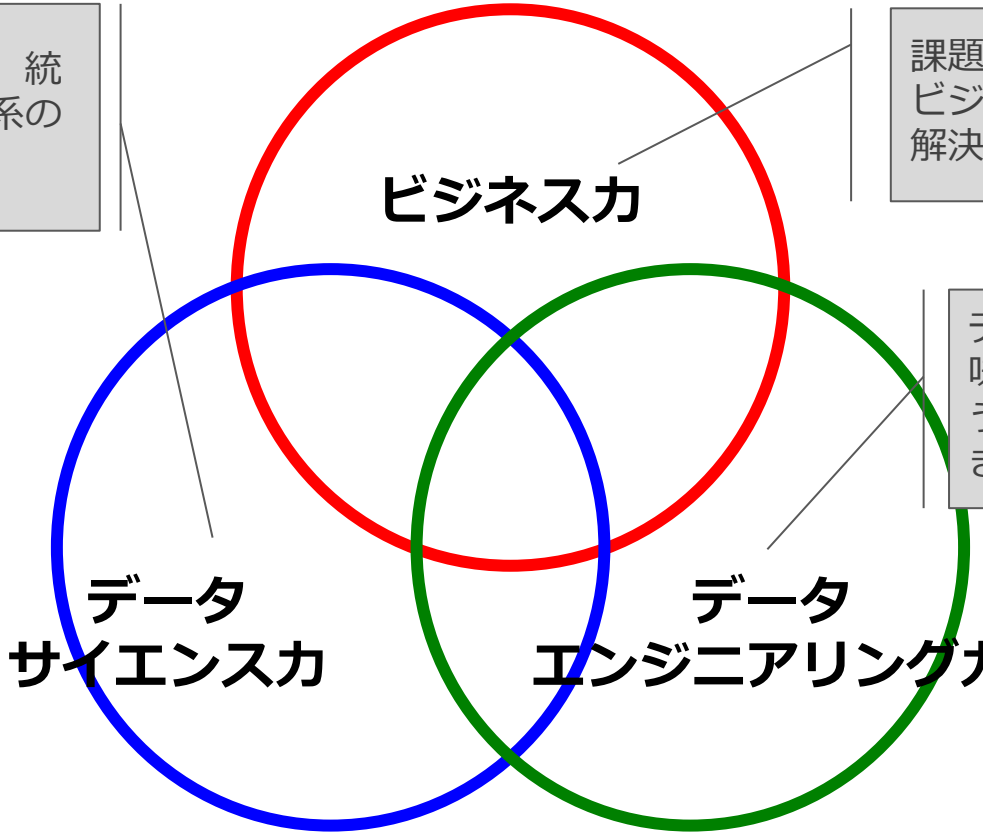
データ分析

プログラム

セクシー

データサイエンティストとは？

情報処理、人工知能、統計学などの情報科学系の知恵を理解し使う力



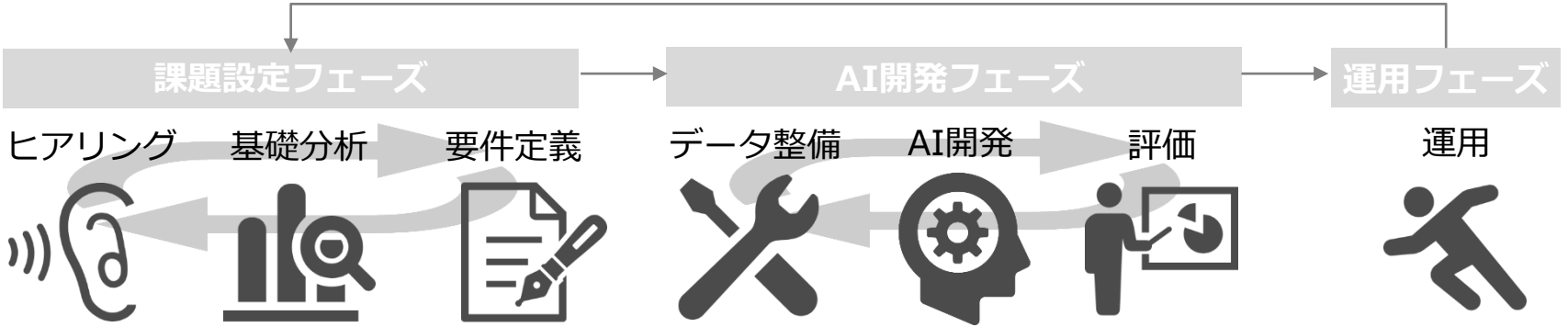
課題背景を理解した上でビジネス課題を整理し解決する力

データサイエンスを意味のある形に使えるようにし、実装、運用できるようにする力

【引用】データサイエンティストに求められるスキルセット　データサイエンティスト協会

データサイエンティストの仕事内容

▼作業内容（例）



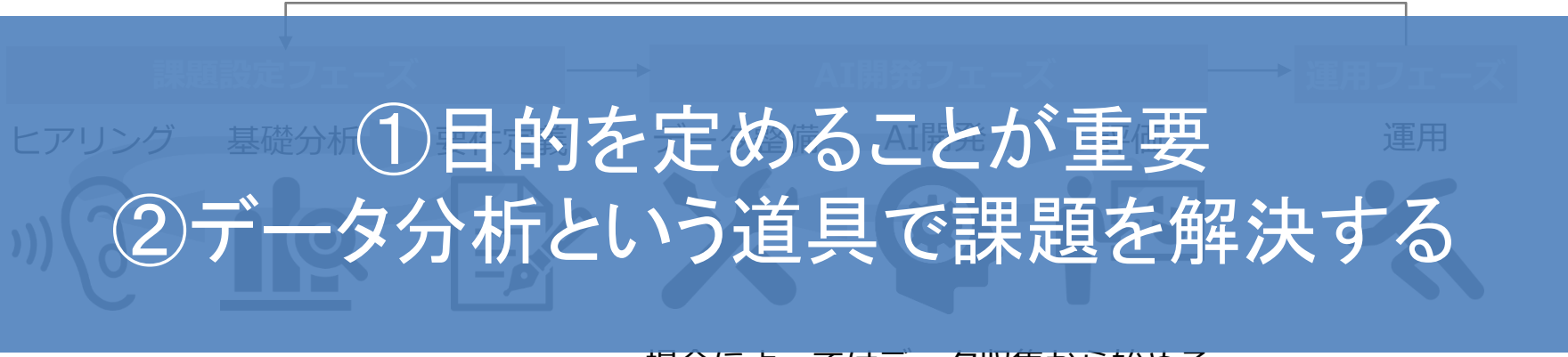
AIを使い、何を実現したいのか？
現在のデータでそれは実現可能か？
具体的に何を作ればいいのか？

場合によってはデータ収集から始める
最適なアルゴリズム選択とチューニング
どの程度のパフォーマンスがでるか？
現場での運用デザイン

現場で運用する
データを蓄積する
運用評価をする

データサイエンティストの仕事内容

▼作業内容（例）



①目的を定めることが重要
②データ分析という道具で課題を解決する

AIを使い、何を実現したいのか？
現在のデータでそれは実現可能か？
具体的に何を作ればいいのか？

場合によってはデータ収集から始める
最適なアルゴリズム選択とチューニング
どの程度のパフォーマンスがでるか？
現場での運用デザイン

現場で運用する
データを蓄積する
運用評価をする

データサイエンティストの道具

▼分析ツール・プログラム言語

▼機械学習フレームワーク

Pythonとは？

「読みやすく書きやすい」を目指したプログラム言語

- OS（windows, mac os等）が異なっていても比較的容易に動かせる
- データ分析に必要なライブラリが豊富であり、R言語と同じようにデータ分析業務で好まれる
- 多くの機械学習フレームワークと互換性がある

※ライブラリとは便利なプログラムをまとめたもの

「読みやすく書きやすい」とはどういうこと？

- Pythonは、文法が非常にシンプルで、書き方が限られるため、誰が書いても似たプログラムになりやすい (=読みやすい)
- また他言語に比べて、プログラムの記述量が非常に少なくすむ（=書きやすい）

<サンプルプログラム> ※PythonとJavaの比較
①画面上に「Hello world!」と表示させるプログラム
②samle.txtというファイルから、データを1行ずつ読み込んで画面上に表示させるプログラム

```python
print('Hello World!')
```
Python（①）

```python
with open('sample.txt', 'r') as f:
    for s in f.readlines():
        print(s.strip())
```
Python（②）

```java
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```
Java（①）

```java
import java.io.BufferedReader;
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;

public class SampleJava {
    public static void main(String[] args) {
        Path path = Paths.get("sample.txt");
        try(BufferedReader br = Files.newBufferedReader
(path)){
            String str = "";
            while((str = br.readLine()) != null) {
                System.out.println(str);
            }
        } catch(IOException e) {
            e.printStackTrace();
        }
    }
}
```
Java（②）

ライブラリがあると何が良い？

- ライブラリとは、プログラムを実装する上で便利な部品をまとめたもの
- 通常、開発者や分析者はライブラリを利用することで、さらに簡略な記述でプログラムを書くことができる
- また、ほとんどがOSS（オープンソースソフトウェア）であり、利用目的を問わず、自由に利用ができる

▼ライブラリを使わない場合

```
def compute_cost(X, y, theta):
    return np.sum((np.dot(X, theta) - y) ** 2) / 2 / len(y)

def gradient_descent(X, y, theta, alpha, num_iters):
    J_history = []
    theta_out = theta
    for i in range(num_iters):
        theta_out -= alpha / len(y) * np.dot(X.T, np.dot(X, theta_out) - y)
        J_history.append(compute_cost(X, y, theta_out))
    return theta_out, J_history


alpha = 0.01
iters = 5000
X2 = np.array((np.ones(len(y)), X)).T
theta = np.zeros(2)
g,cost = gradient_descent(X2, y, theta, alpha, iters)
```

▼ライブラリを使う場合

```
model = LR()
model.fit(X.reshape(-1,1),y)
```

同じ動作のプログラムでも
コードが短く済み、実行時間も高速

現在最も人気のプログラミング言語

- 2021年10月、プログラミング言語の人気・話題性を示すランキング（TIOBE PCI）でPythonが1位に
- C言語、Java以外の言語が1位になるのは20年以上の歴史で初（！）
- TIOBE CEO「JavaとCの長年の覇権は終わり、20年以上ぶりに新しいリーダーが誕生した」

TIOBE Index for October 2021

October Headline: Python programming language number 1!

For the first time in more than 20 years we have a new leader of the pack: the Python programming language. The long-standing hegemony of Java and C is over. Python, which started as a simple scripting language, as an alternative to Perl, has become mature. Its ease of learning, its huge amount of libraries, and its widespread use in all kinds of domains, has made it the most popular programming language of today. Congratulations Guido van Rossum! Proficiat! -- *Paul Jansen CEO TIOBE Software*

The TIOBE Programming Community index is an indicator of the popularity of programming languages. The index is updated once a month. The ratings are based on the number of skilled engineers world-wide, courses and third party vendors. Popular search engines such as Google, Bing, Yahoo!, Wikipedia, Amazon, YouTube and Baidu are used to calculate the ratings. It is important to note that the TIOBE index is not about the *best* programming language or the language in which *most lines of code* have been written.

The index can be used to check whether your programming skills are still up to date or to make a strategic decision about what programming language should be adopted when starting to build a new software system. The definition of the TIOBE index can be found here.

Oct 2021	Oct 2020	Change	Programming Language	Ratings	Change
1	3	↑	Python	11.27%	-0.00%
2	1	↓	C	11.16%	-5.79%
3	2	↓	Java	10.46%	-2.11%
4	4		C++	7.50%	+0.57%
5	5		C#	5.26%	+1.10%
6	6		Visual Basic	5.24%	+1.27%
7	7		JavaScript	2.19%	+0.05%
8	10	↑	SQL	2.17%	+0.61%
9	8	↓	PHP	2.10%	+0.01%
10	17	⇈	Assembly language	2.06%	+0.99%

[出典] https://www.tiobe.com/tiobe-index/

なぜPythonが人気なのか？

- 特徴①「読みやすく書きやすい」
  - 入門言語として使いやすく、利用者の間口が広い
- 特徴②「OSが異なっていても動かせる」
  - 便利なインストーラー（Anaconda）や近年はクラウドの実行環境（Google Colaboratory等）もあり、環境構築やライセンスに悩まずに使いやすい
- 特徴③「ライブラリが豊富」
  - 特に機械学習やデータ分析に特化したライブラリが整備されており、近年のAIブームにも乗って人気が急上昇
- 特徴④「汎用性が高い」
  - 特にAIや機械学習で注目を集めてはいるが、それ以外にもアプリケーション開発やデータ収集（Webスクレイピング）などにも使うことができる
  - 1つの言語でやりたい事をシームレスに実現できるのも人気の理由

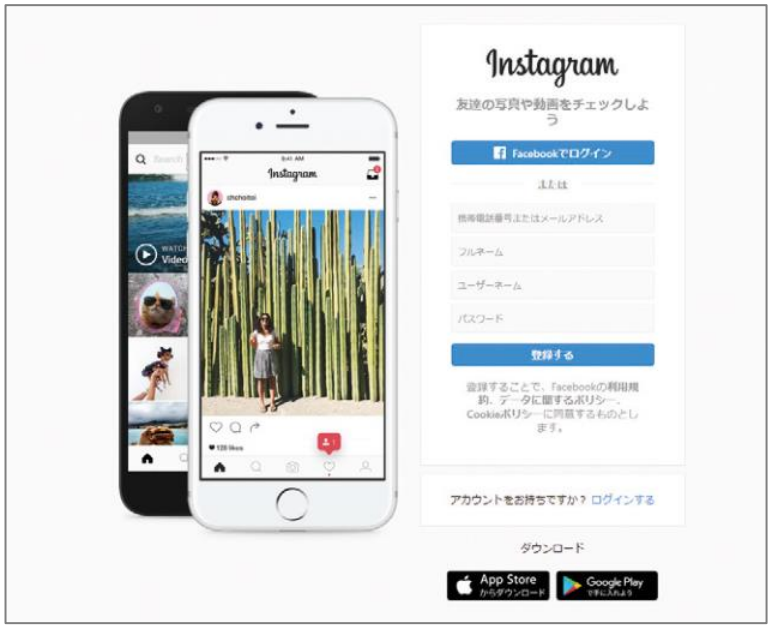
Pythonでできること

- アプリケーション開発（Webアプリ, デスクトップアプリ）
- タスクの自動化
- Web情報の自動収集（スクレイピング）
- データ分析・機械学習

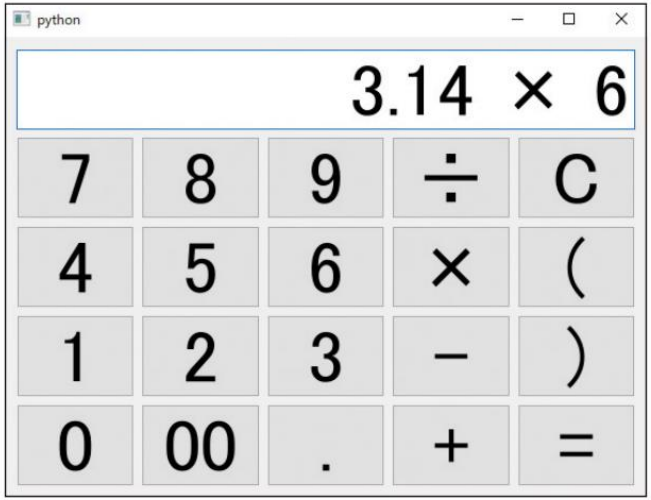
Pythonでできること

アプリケーション開発（Webアプリ, デスクトップアプリ）

▼ Pythonが使われているWebサービス（Instagram）

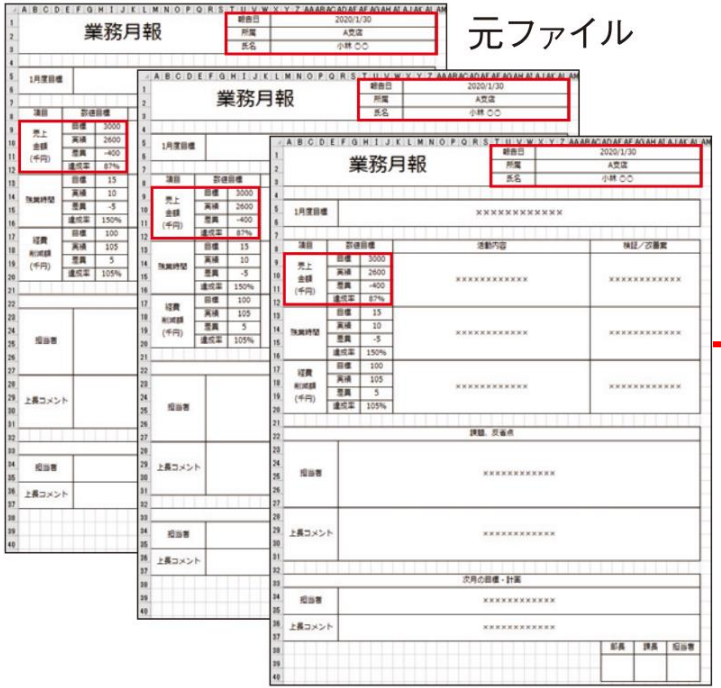


▼ Pythonで作られたGUI（電卓アプリ）



Pythonでできること

タスクの自動化（例：Excelデータの整理・集計）



元ファイル

バラバラに散らばったExcelファイルを
まとめて一覧化・集計を行う など

A1 報告日

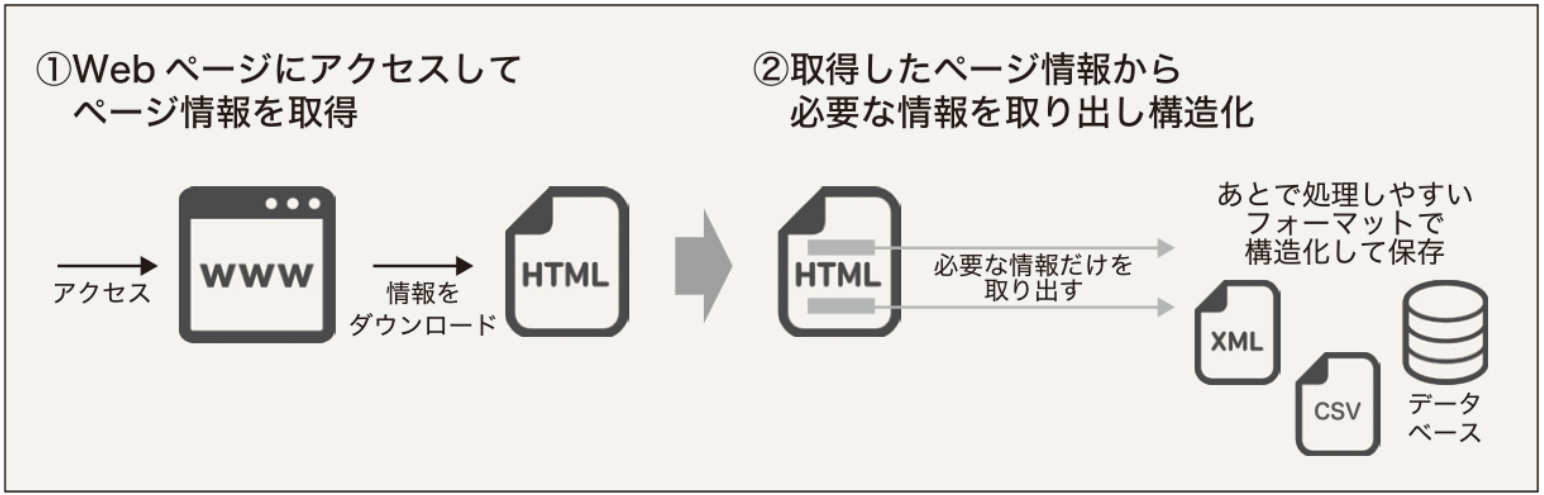
	A	B	C	D	E	F
1	報告日	支店	担当者	売上目標(千円)	売上実績(千円)	差異(千円)
2	2020年01月30日	A支店	小林 ○○	3000	2600	-400
3	2020年01月31日	A支店	新谷 ○○	3300	3250	-50
4	2020年01月31日	A支店	竹田 ○○	3000	3100	100
5	2020年01月29日	B支店	平山 ○○	2200	2100	-100
6	2020年01月31日	B支店	金城 ○○	2200	1980	-220
7	2020年01月31日	C支店	丸高 ○○	3100	3700	600
8	2020年01月30日	C支店	小田島 ○○	3100	3500	400
9	2020年01月31日	C支店	青木 ○○	3100	3350	250
10	2020年01月31日	D支店	佐々木 ○○	2500	2600	100
11	2020年01月31日	D支店	島村 ○○	2500	2800	300
12	2020年01月31日	E支店	畑中 ○○	1900	1920	20
13	2020年01月31日	E支店	米沢 ○○	1900	1700	-200
14						
15						
16						

月報一覧

Pythonでできること

Web情報の自動収集（スクレイピング）

- スクレイピングとは ①Webページを巡回して情報を取得（クローリング）し、②取得した情報から必要な情報を抽出（パース）する行為のこと
- 例えば、あるオンラインショップの製品情報・価格情報の一覧を取得する など

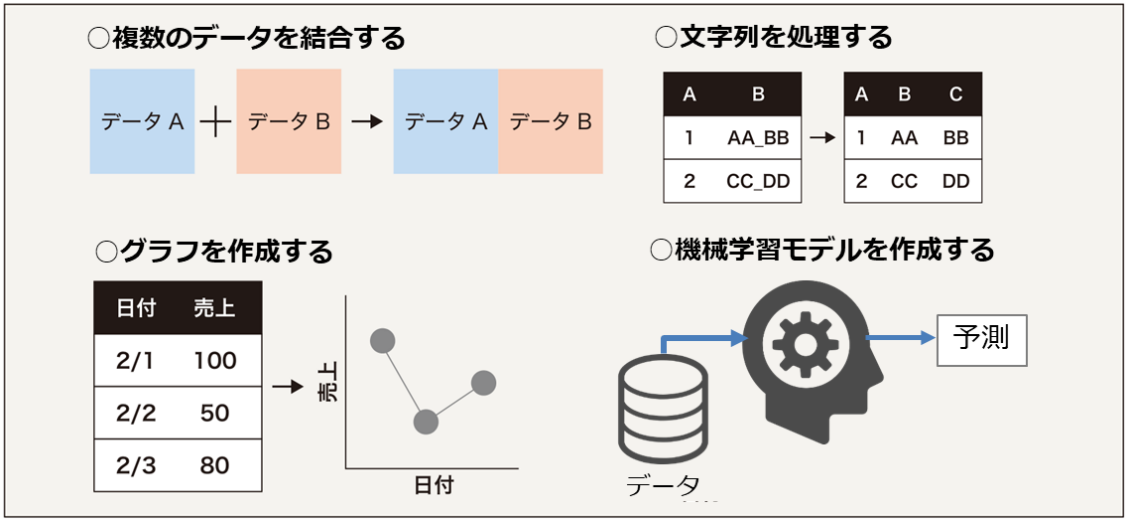


※ただし、スクレイピングを禁止しているWebサイトもあるので注意が必要です

Pythonでできること

データ分析・機械学習

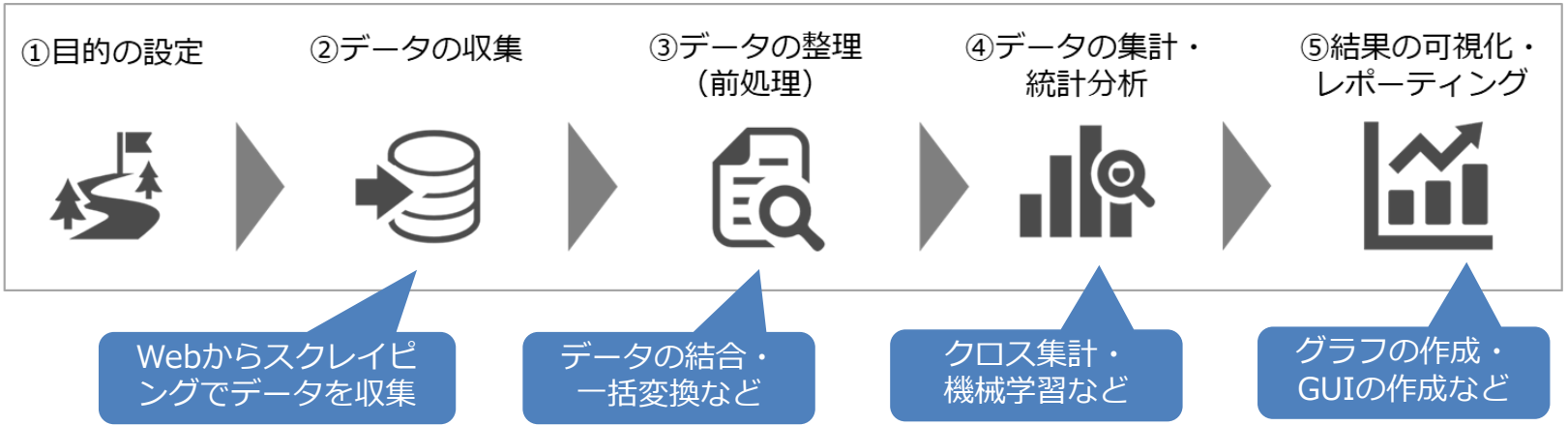
- データ分析では、グラフの作成や機械学習モデルの作成を行うだけでなく、事前にデータを分析しやすいように加工する「前処理」という作業も必要になる
- 例えば、複数のデータを1つに結合する、一括でデータの値を変換する　など



データ分析プロジェクトで威力を発揮

特にデータ分析プロジェクトでは、②データの収集～③整理～④集計・分析～⑤可視化まで、Python1つでシームレスに実行できるのが強み

▼一般的なデータ分析の流れ



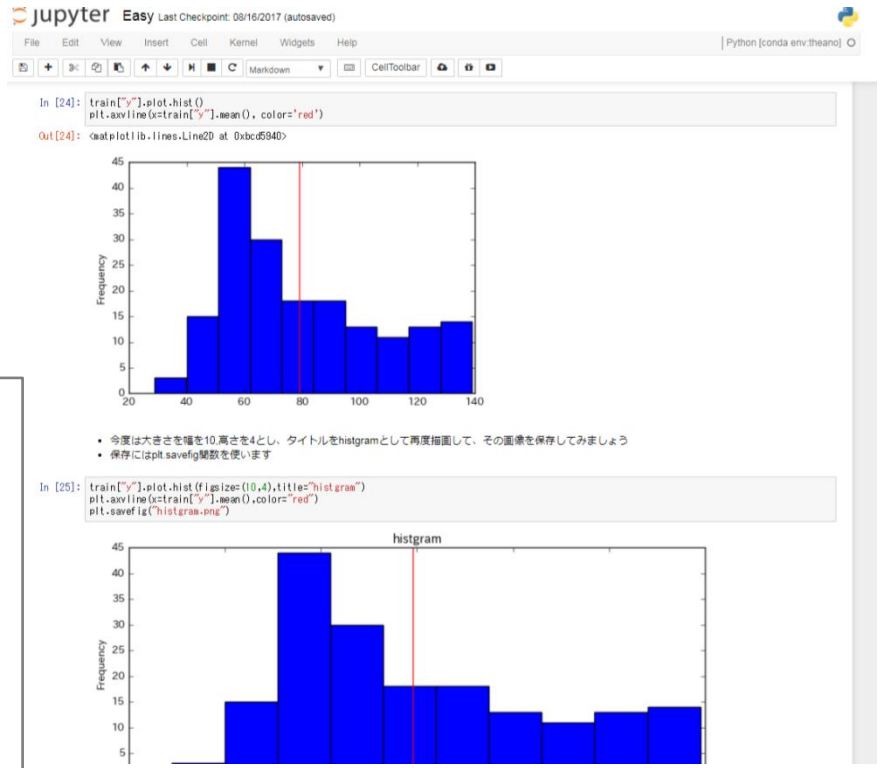
ここまでのまとめ

- Pythonとは「読みやすく書きやすい」人気のプログラミング言語
  - 特に初学者の入門言語として適している
- Pythonでできること
  - アプリ開発から機械学習まで汎用的な用途で使用可能
  - データ分析プロジェクトの一連の流れをPython1つでシームレスに実行可能

＜後半のお話では＞

- Pythonの基礎（文法的なお話）
- SIGNATE Questを活用してPythonや機械学習を学ぶ方法の手引き

Jupyter Notebookとは？



python等のプログラムを書いて簡単に動かすことができるツール

- ブラウザ上で動作
- 対話型でプログラミング可能
- 可視化やメモが簡単に

Google Colaboratoryとは？

Googleが提供するnotebook

・環境設定不要でインターネットがあれば使える

・GPUが無料で利用できる

・共有が簡単にできる

午後の演習について

- **Slackを使います**
  - まだサインインできていない人いますか？
  - スマホアプリもあるのでインストールしてみてください

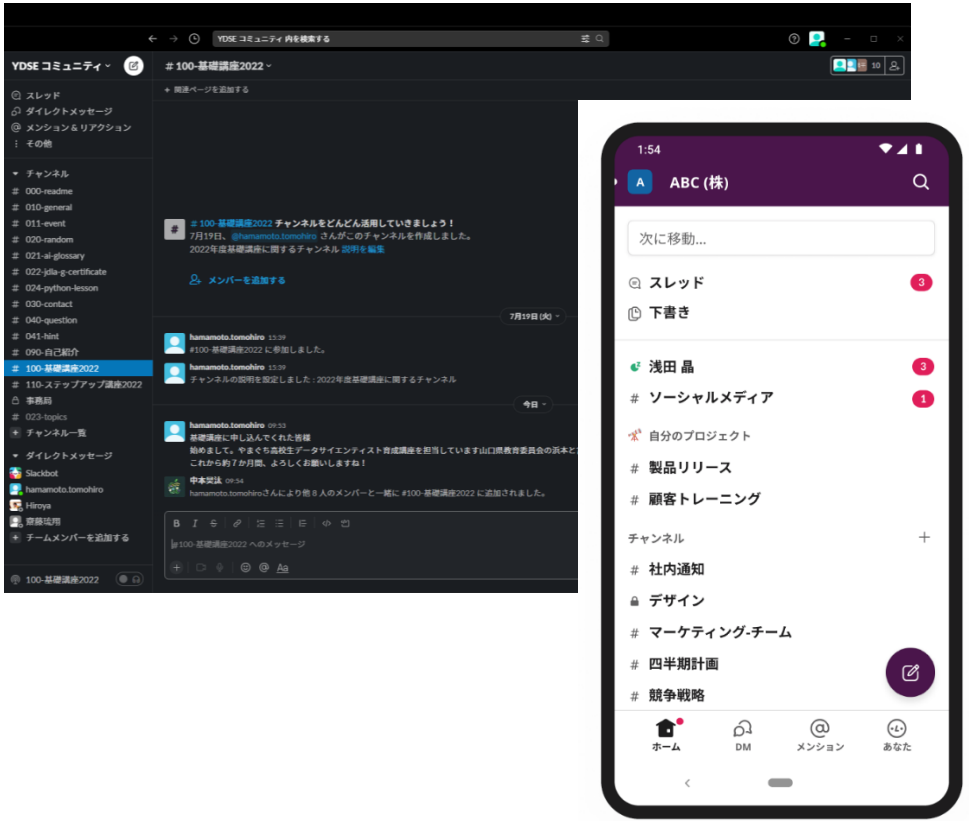
- **SIGNATE Cloudを使います**
  - まだサインインできていない人いますか？
  - https://biz.quest.signate.jp/biz_users/sign_in

Slackの使い方

Slackとは？

What is Slack？

Slack は、人々をそれぞれが必要とする情報につなげる、ビジネス用のメッセージングアプリです。Slack を使うことで、メンバーが 1 つの場所に集まり、チームが一体となって働くことができ、組織のコミュニケーションの方法が変わります。



Slackとは？

Slackとは？

Slackとは？

スタンプ

Slackにおいて最も重要な機能だと思っています。LINEのように既読とはつかないため、このメッセージを見たのかどうかや、それに対する反応は**自発的**に行う必要があります。

いろんな種類のスタンプがあるので、投稿があったら必ず反応しよう！

Slackとは？

メンション

誰かに対するメッセージはメンションを付けることで、相手にメッセージがあったことをお知らせできます。

特定の誰かに向けたメッセージであれば必ずメンションを使いましょう。

```
@channel
  →チャンネルの人全員に通知

@名前
  →名前の人に通知
```

Slackの書き方

マークアップ

デスクトップで Slack を使っている場合は、マークアップを使ってメッセージの書式を設定することもできます。

書式設定	説明
Bold	テキストをアスタリスクで囲みます： *対象のテキスト*
Italicize	テキストをアンダースコアで囲みます： _対象のテキスト_
~~Strikethrough~~	テキストの前後をチルダで囲みます： ~対象のテキスト~
`Code`	テキストをバッククォートで囲みます： `対象のテキスト`
Block quote	テキストの前に山括弧を追加します： >対象のテキスト
`Code block`	テキストの前に 3 つのバッククォートを追加します： ```対象のテキスト
1. Ordered list	テキストの前に 1 とピリオドを追加して、スペース キーを押します： 1. 対象のテキスト 行をインデントするには、Tab を押します。
• Bulleted list	テキストの前にアスタリスクを追加して、スペース キーを押します： * 対象のテキスト 行をインデントするには、Tab を押します。

よく使うもの

`強調`

`はshift+@

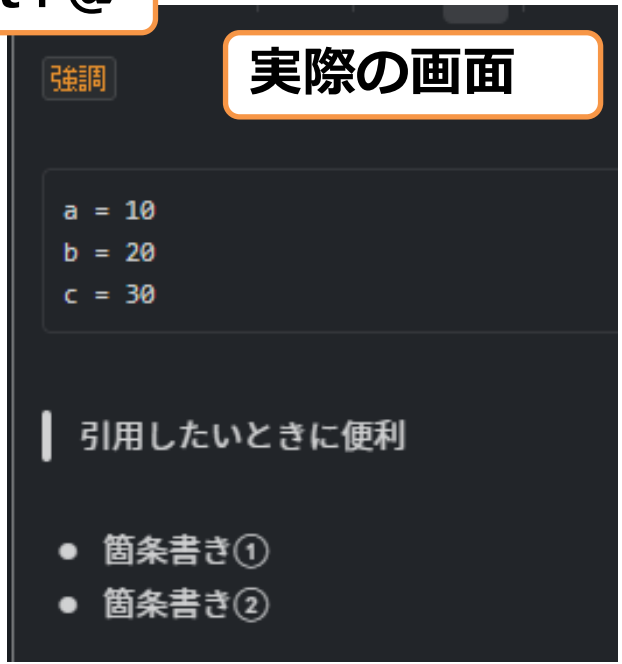
```
コードブロック
```

>引用

- 箇条書き①
- 箇条書き②

-と箇の間にスペースが必要

実際の画面



Slackとは？

文字のコミュニケーションは気を付けるべし。

文字のコミュニケーションでは皆さんご存じの通り、感情がわからないため、知らず知らずに相手のことを不愉快にしてしまうこともあります。

その為、投稿メッセージは
・いろんな人が見ていることを意識する
・なるべく丁寧な言葉遣いを選ぶ
・スタンプをうまく使う

を心掛けたいですね。

これからやっていただきたいこと

① 「100-基礎講座2022」チャンネルを開く
② 以下の投稿を行ってください。
・名前：
・学校：
・学年：
・部活：
・趣味／最近はまっていること：
・行ってみたいところとその理由：
・自分を動物に例えると：

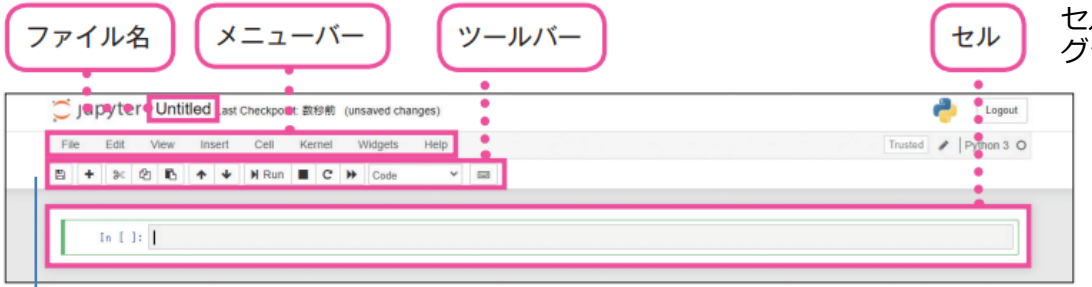
③ 他の方の投稿にスタンプを押す
④ （なるべく）**接点がない方5人**の投稿に対し、スレッドでメンションを付けて何か質問をする
⑤ 自分に対する質問に質問者へのメンションを付けて、スレッドで回答する
⑥ 回答に対し、スタンプやメッセージで反応する

こんな感じ

Jupyter Notebookの使い方

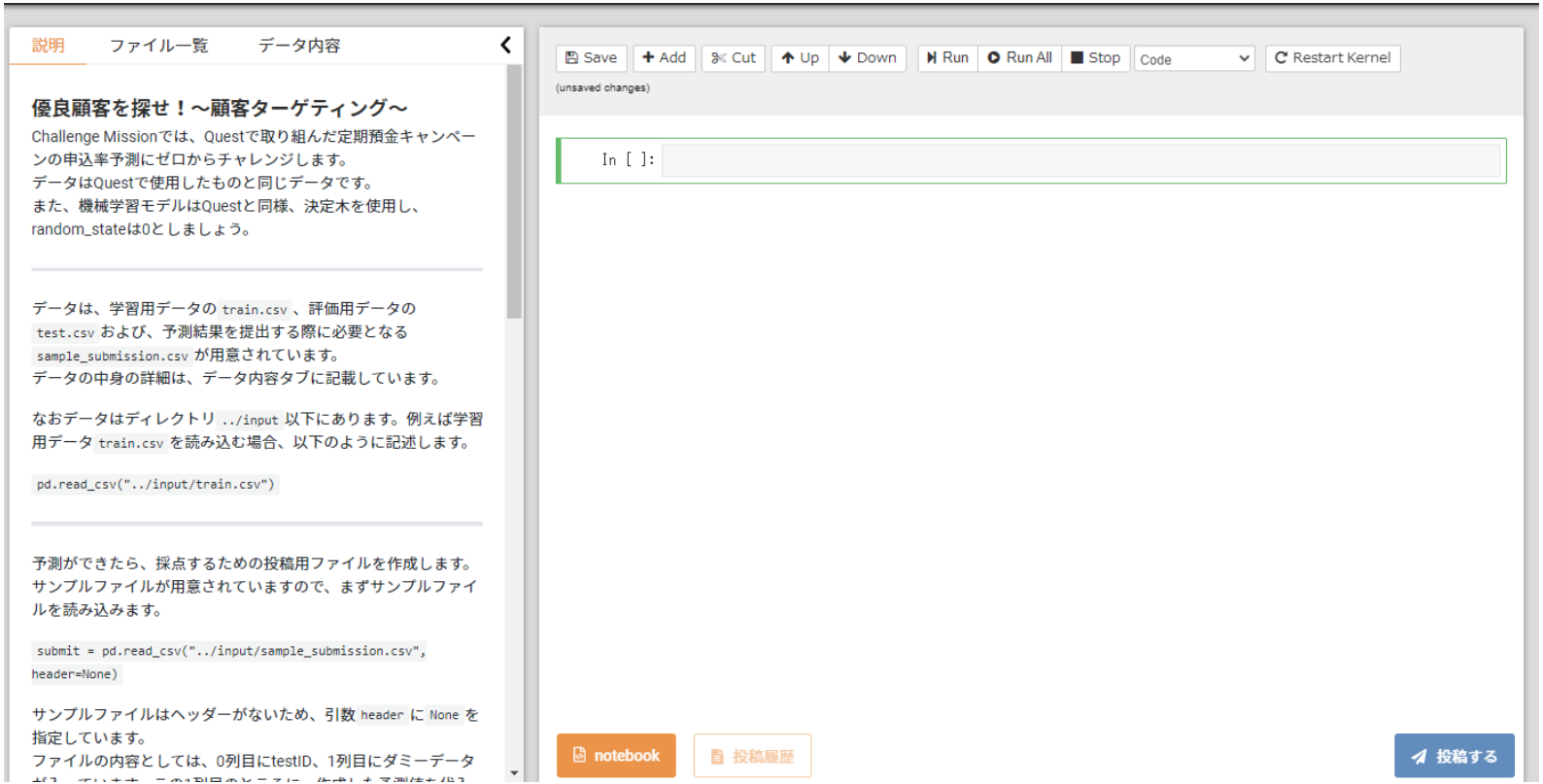
Jupyter Notebook

NotebookのUIは次のようになっています。



よく使うツールバーのアイコン

Challenge MissionのUI

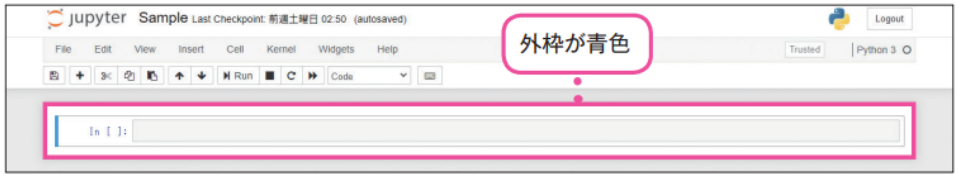


Jupyter Notebook

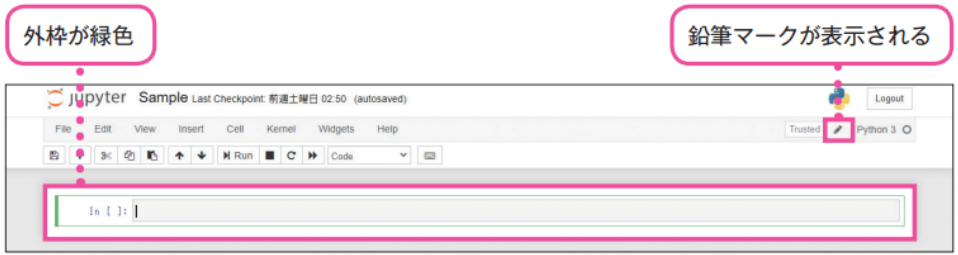
Notebookを利用するうえで重要になるのは「セル」です。
In[]の右にある入力欄を「セル」と呼び、このセルには「コマンドモード」と「編集モード」というモードが2種類あります。

「コマンドモード」・・・セルを増やしたり減らしたり、操作するときのモード。外枠が青色。
「編集モード」・・・プログラムを入力するときのモード。外枠が緑色＋鉛筆マークが表示。

●コマンドモード

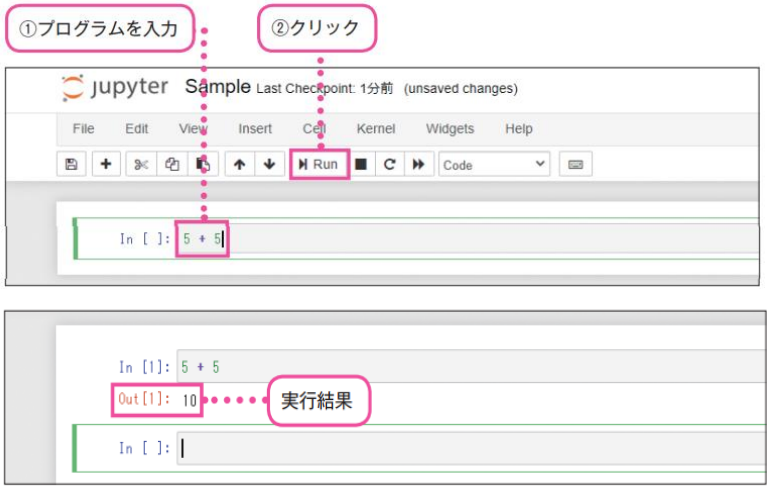


●編集モード



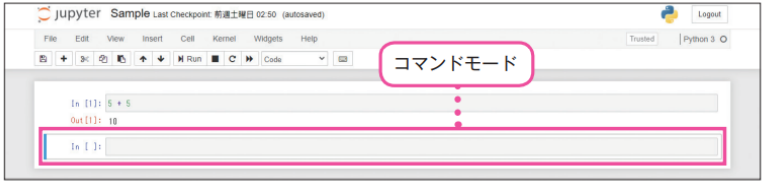
Jupyter Notebook

実際に動作の確認をしてみます。
セルをクリックすると、カーソルが表示され「編集モード」に切り替わります。
セル内に「5+5」を入力し、プログラムを実行してみてください。
プログラムの実行はツールバーの「Run」アイコンか、Shiftキーを押しながらEnterキーを押します。
後者の実行のショートカットキーは頻繁に利用するので、覚えておくと良いでしょう。



プログラムを実行すると、Out[1]の右側に実行結果が表示される。In[1]やOut[1]の数字はプログラムが何回目に実行されたかの履歴番号を表す。

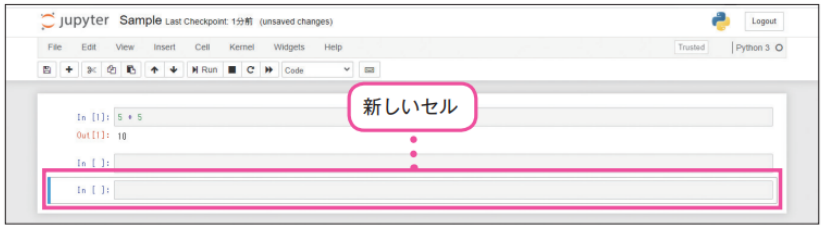
実行後、自動的に新規セルが作成されます。
この状態で入力欄以外をクリックするか、Escキーを押すと、コマンドモードに切り替わります



Jupyter Notebook

コマンドモードの状態ではセルを増やしたり減らしたりすることができます。
「B」キーを1回押すとセルが増え、「D」キーを2回押すとセルが削除されます。
「H」キーを押すと、ショートカットキーの一覧が表示されます。

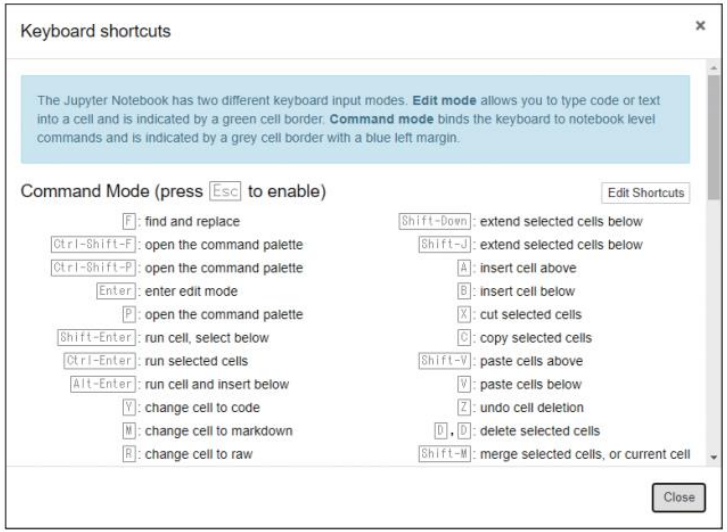
●図3-14　Bキーによる新規セルの作成



今度はDキーを2回押すと、セルが1つ削除されます。

●図3-15　Dキーの2回押しによるセルの削除

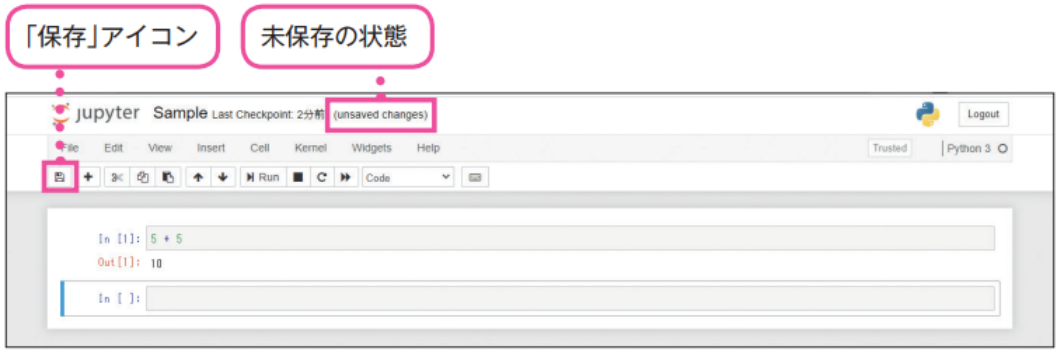
●図3-16　Hキーで表示されるショートカットキーの一覧



Jupyter Notebook

Notebookファイルの保存は、「保存」アイコンをクリックします。
Notebookには自動保存機能もあるため、画面の上部に「autosaved」が表示されていれば手動保存は必要ありません。
未保存の場合には、「unsaved changes」と表示されますので、この時は手動保存しましょう。

なお、ブラウザとターミナルを終了すればJupyter Notebookは終了します。
Notebookファイルはプログラムのソースコードと実行結果については保存されますが、**一度終了したnotebookを再開したいときには、最初からプログラムを実行しなおす必要があります**。つまり、Notebookを終了すると、プログラムのソースコードや実行結果は残っていますが、ライブラリの宣言や変数の値はすべてリセットされてますので、再度の実行が必要です。その点についてよく注意しましょう。



Jupyter Notebook

覚えてほしいショートカットキー

ショートカット内容	利用できるモード	ショートカットキー
セルを実行する	編集モード	Shift + Enter
編集モードからコマンドモードに切り替える	編集モード	ESC
コマンドモードから編集モードに切り替える	コマンドモード	Enter
セルを増やす	コマンドモード	B
セルを削除する	コマンドモード	D（2回押し）
削除したセルを元に戻す	コマンドモード	Z
ショートカットキー一覧を表示する	コマンドモード	H

Pythonのキホンの「キ」

Pythonのキホンの「キ」

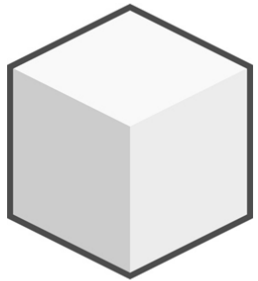
1 **まず型を覚えるべし**

2 エラーから逃げるべからず

3 ググる習慣を身につけるべし

変数

プログラミングではデータを**変数**に代入して利用します



数字：5
文字列：”あいうえお”
データフレーム：

	date	Count
0	4/1	20
1	4/2	30

<E.g.>

value = 5

char = “テスト”

train = pd.read_csv(“train.csv”)

変数

例. お釣りの計算 (支払い金額 – 商品の価格 = お釣り)

▼ 変数を使わない場合

500 – 100 = 400円
500 – 450 = 50円
500 – 200 = 300円

▼ 変数を使う場合

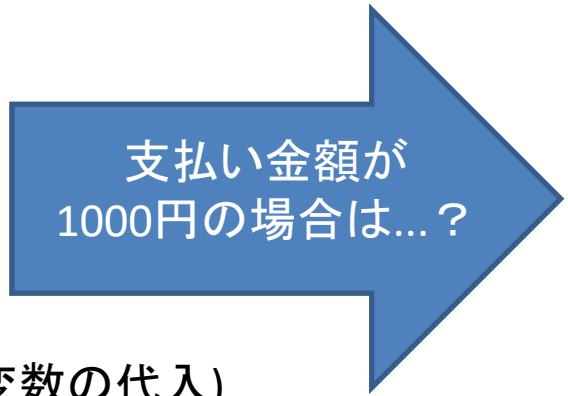
変数aを500とする。(変数の代入)
a – 100 = 400円
a – 450 = 50円
a – 200 = 300円

変数

例. お釣りの計算 (支払い金額 – 商品の価格 = お釣り)

▼ 変数を使わない場合

500 – 100 = 400円
500 – 450 = 50円
500 – 200 = 300円



▼ 変数を使う場合

変数aを500とする。(変数の代入)
a – 100 = 400円
a – 450 = 50円
a – 200 = 300円

変数

例. お釣りの計算 (支払い金額 – 商品の価格 = お釣り)

▼ 変数を使わない場合

500 – 100 = 400円
500 – 450 = 50円
500 – 200 = 300円

支払い金額が
1000円の場合は…？

1000 – 100 = 900円
1000 – 450 = 550円
1000 – 200 = 800円

▼ 変数を使う場合

変数aを500とする。(変数の代入)
a – 100 = 400円
a – 450 = 50円
a – 200 = 300円

変数aを**1000**とする。
a – 100 = 900円
a – 450 = 550円
a – 200 = 800円

変数

例. お釣りの計算 (支払い金額 – 商品の価格 = お釣り)

▼ 変数を使わない場合

500 – 100 = 400円
500 – 450 = 50円
500 – 200 = 300円

1000 – 100 = 900円
1000 – 450 = 550円
1000 – 200 = 800円

1000円の場合は…？

変数を使ったほうが変更箇所が少なく済む！

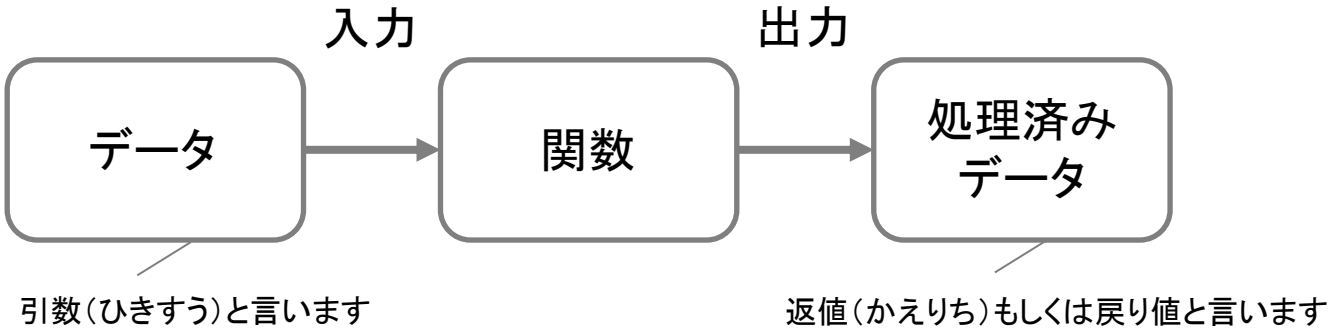
▼ 変数を使う場合

変数aを500とする。(変数の代入)
a – 100 = 400円
a – 450 = 50円
a – 200 = 300円

変数aを**1000**とする。
a – 100 = 900円
a – 450 = 550円
a – 200 = 800円

関数

一連の操作や処理をまとめたものを**関数**と言い、頻繁に利用します
⇒Excelで言うSUM関数などと同じ



関数名()
関数名(引数)
関数名(引数1, 引数2)

関数は必ずカッコが必要
引数がない場合もある

関数

例. お釣りの計算 (支払い金額 − 商品の価格 = お釣り)

▼ 関数の定義

```
def お釣り計算(支払い金額, 商品の価格):
    お釣り = 支払い金額 - 商品の価格
    return お釣り
```

▼ 関数の実行

実行結果

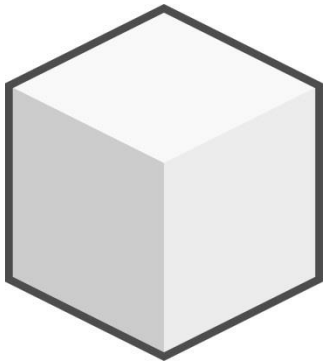
```
お釣り計算(500, 100)
```
→ 400

```
お釣り計算(500, 450)
```
→ 50

※ 関数内で行う処理がどれだけ長くなっても、
実行時のプログラムは1行で済む。

オブジェクトとデータ型

Pythonではデータのことを“**オブジェクト**”と言います。
それぞれの“**オブジェクト**”は、特定の”**データ型**”に属しています。



整数　int型　(例. 1, 2, 3, ...)
浮動小数点数　float型　(例. 1.5, 3.14, ...)
文字列　str型　(例. “a”, “abc”, ...)
表　DataFrame型 (例. ↓の表)

Out[4]:

	ID	model_year	weight	color	tire	kpl
0	0	76	2120	silver	pirelli	34.6
1	1	76	2579	yellow	michelin	24.0
2	2	78	2560	blue	goodyear	30.3
3	3	78	2068	yellow	pirelli	39.2
4	4	71	2385	silver	goodyear	23.5

オブジェクトの型によって、
「付属情報(属性)」「できること(関数)」が異なる

オブジェクトと型

“**オブジェクト**”を物理的なものに当てはめて考えると...

	▼「人間」型	▼「自動車」型
		
	とある人物Aさん	Aさんが所持する自動車B
付属情報)	高さ → 170.0cm (身長) 血液型 → O型	高さ → 205.0cm (車高) 血液型 → ×
できること)	食事を摂る → ○	食事を摂る → ×

“.”(ドット)記号

オブジェクトを操作する際に利用するのが**“.”（ドット）**

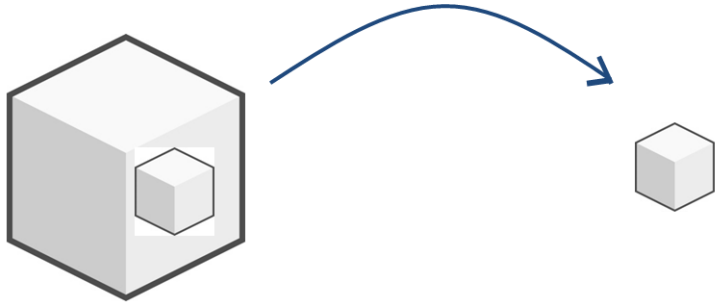
変数trainはDataFrame型のオブジェクト。
DataFrame型のオブジェクトは、
<u>shapeという名前で行数と列数を保持</u>している。

Out[4]:

	ID	model_year	weight	color	tire	kpl
0	0	76	2120	silver	pirelli	34.6
1	1	76	2579	yellow	michelin	24.0
2	2	78	2560	blue	goodyear	30.3
3	3	78	2068	yellow	pirelli	39.2
4	4	71	2385	silver	goodyear	23.5

行数

列数



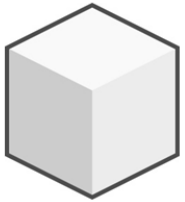
<E.g.> train.shape

ドット

オブジェクトの関数（メソッド）を使うとき

オブジェクトを操作する際に利用するのが**“.”（ドット）**

オブジェクトによって使えるものは違います



オブジェクト . 関数()

<E.g.>

train.plot()

train.plot(option)

train.plot(option1 , option2)

2つ以上の時はカンマ（, ）を使う

オブジェクトと型

“**オブジェクト**”を物理的なものに当てはめて考えると…

	▼「人間」型	▼「自動車」型
		
	とある人物Aさん	Aさんが所持する自動車B
付属情報)	A.height → 170.0 (身長) A.blood_type → O型	B.height → 205.0 (車高) B.blood_type → ×
できること)	A.eat() → 食べる	B.eat() → × (自動車は食べない)

ライブラリ

“ライブラリ”とは、特定の用途に応じて便利な処理をまとめたもの

※ イメージ

ライブラリ
(モジュール)　≒

道具箱

ライブラリは、便利な道具。
上手く活用することで、プログラムを効率的に記述することができる。

よく使うライブラリー覧

Pythonでデータ分析を行う上で欠かせないライブラリの一覧

- **Pandas**
  - 表形式(2次元)データの解析が得意なライブラリ
- **NumPy**
  - 多次元配列の数値計算が得意なライブラリ
- **Matplotlib**
  - グラフの可視化ライブラリ

※ SIGNATE Cloudでは、「Gym Pandas入門道場」のような講座内でこれらのライブラリの使い方を個別に学ぶことができます。

データ分析の型

どんな分析においてもPythonにおけるデータ分析の流れは基本同じです

⇒pandas, numpy, matplotlib, sklearn

⇒pandasのread_csv

⇒pandasの関数、matplotlib

⇒pandasの関数、matplotlib、numpy、sklearn

データ分析の型

どんな分析においてもPythonにおけるデータ分析の流れは基本同じです

実はPythonというよりライブラリの使い方を覚えることが重要

絶対に覚えてほしいシリーズ①

▶ライブラリのインポート

```
# インポートのキホン
import pandas

# 省略して使うのが慣習的
import pandas as pd

# matplotlibのインポート（下記はどっちも同じです）
from matplotlib import pyplot as plt
import matplotlib.pyplot as plt
```

絶対に覚えてほしいシリーズ②

▶データを読み込むpandasの「read_csv」関数

```
import pandas as pd

# csvファイル
data = pd.read_csv("sample.csv")

# indexなし、headerなし
data = pd.read_csv("sample.csv", index=None, header=None)

# tsvファイル
data = pd.read_csv("sample.tsv", sep="¥t")
```

絶対に覚えてほしいシリーズ③

▶データを中身を見るpandasの「head」関数

```
import pandas as pd

# データ読み込み
data = pd.read_csv("sample.csv")

# 行数列数の確認
data.head()
```

Out[4]:

	ID	model_year	weight	color	tire	kpl
0	0	76	2120	silver	pirelli	34.6
1	1	76	2579	yellow	michelin	24.0
2	2	78	2560	blue	goodyear	30.3
3	3	78	2068	yellow	pirelli	39.2
4	4	71	2385	silver	goodyear	23.5

絶対に覚えてほしいシリーズ④

▶データの行数・列数を確認するshape（関数ではないことに注意）

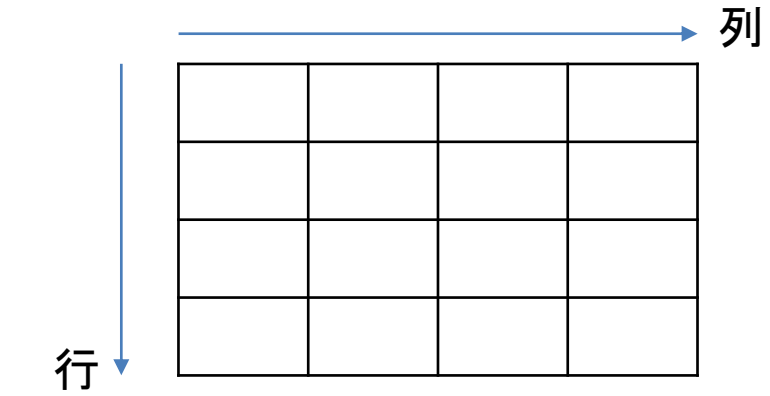
```
import pandas as pd

# データ読み込み
data = pd.read_csv("sample.csv")

# 行数列数の確認
data.shape
```

```
Out[3]:

(1990, 6)
```



絶対に覚えてほしいシリーズ⑤

▶pandasのDataFrameから列（カラム）を抽出

```
import pandas as pd

# データ読み込み
data = pd.read_csv("sample.csv")

# 列Aを指定
data["A"]

# 複数列を指定
data[["C", "D"]]
```

data[“A”]　　data[[“C”,”D”]]

A	B	C	D
1.0	2.0	1.5	4.2
2.0	1.0	3.5	2.1
3.0	2.0	7.5	0.5

Pythonのキホンの「キ」

1 まず型を覚えるべし

2 **エラーから逃げるべからず**

3 ググる習慣を身につけるべし

初学者の壁

初学者の壁

エラーは味方です！

エラーに関する勘違い

こんなにエラーが出るなんて、
自分には才能がないんじゃ。。。

エラーに関する勘違い

こんなにエラーが出るなんて、
自分には才能がないんじゃ。。。

上級者でもエラーを出すのは日常茶飯事！

エラーとの向き合い方

学び始め

なぜエラーが起きているかがわからない…

エラーの原因を調べれば解決できるように

エラー文を見た瞬間に解決方法がわかる

慣れた頃

エラーとの向き合い方

学び始め

なぜエラーが起きているかがわからない…

「エラーを出さない」ではなく、
「エラーを素早く正確に解決する」スキルを身につけよう

エラー文を見た瞬間に解決方法がわかる

慣れた頃

よくあるミス

- **スペルミス**
  - describeをdescriveと書いてしまう
- **大文字・小文字のミス**
  - Falseをfalseと書いてしまう
  - pythonでは大文字と小文字は別物として区別されます
- **カッコの有無・カッコの種類**
  - 関数ではカッコが必要となります
  - head()をheadと書いてしまう
  - {}と[]と()は異なる
- **ドットとカンマを間違える**
  - sort_values(by="y" . ascending=False)と書いてしまう

エラーの見方

```
x = 10
y = 20
print(z)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-35-35cd9fc45517> in <module>
      1 x = 10
      2 y = 20
----> 3 print(z)

NameError: name 'z' is not defined
```

エラーの原因となっているプログラムの場所

エラーの理由

エラー名

代表的なエラー

- ModuleNotFound Error
- Name Error
- Syntax Error
- FileNotFound Error
- Attribute Error

ModuleNotFound Error

```
import pandus
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
<ipython-input-34-d5dd31d7b47c> in <module>
----> 1 import pandus

ModuleNotFoundError: No module named 'pandus'
```

インポートしたいライブラリが見つからない
⇒ライブラリ名にタイプミスがないかを確認しましょう

Name Error

```
x = 10
y = 20
print(z)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-35-35cd9fc45517> in <module>
      1 x = 10
      2 y = 20
----> 3 print(z)

NameError: name 'z' is not defined
```

その変数が見つからない
⇒その変数名のタイプミスがないかを確認しましょう

Syntax Error

```
x = 10
if x > 10
   print(x)
```

```
  File "<ipython-input-36-67b635acfe7f>", line 2
    if x > 10
           ^
SyntaxError: invalid syntax
```

プログラムの文法が間違っている
⇒記号忘れ、その記号であっているか等を確認しましょう

FileNotFound Error

```
import pandas as pd

data = pd.read_csv('sample.csv')
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
<ipython-input-38-b6eb618266ff> in <module>
      1 import pandas as pd
      2
----> 3 data = pd.read_csv('sample.csv')
〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜
FileNotFoundError: [Errno 2] File b'sample.csv' does not exist: b'sample.csv'
```

読み込みたいファイルが見つからない
⇒ファイル名やファイルパスを確認しましょう

Attribute Error

```
i = 100
print(i.split('_'))
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-39-282fe4aa2ba1> in <module>
      1 i = 100
----> 2 print(i.split('_'))

AttributeError: 'int' object has no attribute 'split'
```

ドット演算子でアクセスできない要素・関数を呼び出した
⇒データ型や関数名などを確認しましょう

Pythonのキホンの「キ」

1 まず型を覚えるべし

2 エラーから逃げるべからず

3 **ググる習慣を身につけるべし**

エラーに関する勘違い

何度もやったことがあるのに
またこの処理の書き方を忘れた…

全てのプログラムを覚える必要はない。
調べながらプログラムを書くのが当たり前！

ググる習慣を身につけるべし

・**○○がしたい！**

→「python 対数変換」でググる

・**プログラムの使い方を思い出せない**

→ 「pd.read_csv 引数」でググる

・**エラーが発生**

→ 「SyntaxError: invalid syntax」でググる

・**プログラムの実行結果がおかしい**

→ 「matplotlib 文字化け 四角」

ググる習慣を身につけるべし

・プログラムを覚えられない、すぐ忘れてしまう
・よくわからないエラーが発生
・〇〇がしたい！

検索エンジンを上手に利用しましょう

ライブラリ名 ＋ 作業内容

Python ＋ エラー名、エラー理由

Python ＋ 技術名、やりたいこと

問題解決のためのおすすめサイト

- **公式系**
  - 公式なので、情報が正確。
    (ただし英語かつ文章が堅いので、はじめはやや理解が難しい)
  - 例) Pandas公式サイト、GitHubリポジトリ等の配布元
- **ブログ系**
  - 解決策が具体例とともに提示されており、わかりやすい。
  - 例) 「Qiita」「note.nkmk.me」等
- **Q&A系**
  - 解決策がWeb上で見つからない場合は自分で質問可能。
  - 「Stack Overflow」「teratail」等

次回について

もう少し“本格的”にデータ分析を行います！

次回までにやってきてほしいこと
- AI入門（2h）
- Python入門（8h）

できればやってきてほしい！
- **自動車環境性能の改善（6h）**
- pandas入門道場（2h）

おしまい

Let's Enjoy DataScience!!!