

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

だいたいわかるようになります！

Python

DeepLearning

lightGBM

Seaborn



Boost

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の

新たな社会 "Society 5.0"

超スマート化
???

5.0



1.0
Society 1.0 狩猟



2.0

Society 2.0 農耕

軽工業

- 蒸気機関
- 紡績機

自動化・情報化

- コンピュータ
- インターネット

4.0



Society 4.0 情報

[内閣府作成]



Society 3.0 工業

3.0

重化学工業

- 電力
- 石油
- モーター

出典: Society5.0 科学技術政策 内閣府

Society 5.0

これまでの情報社会(4.0)

Society 5.0



IoTや5Gにより
知識や情報が密に共有

ビッグデータを通じて
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



出典：Society5.0 科学技術政策 内閣府

もうひとつ

DX

DXとは？

Digital Transformation (DX)

企業がビジネス環境の激しい変化に対応し、**データとデジタル技術を活用して**、
顧客や社会のニーズを基に、**製品やサービス、ビジネスモデルを変革するとともに**、
業務そのものや、組織、プロセス、企業文化・風土を変革し、競争上の優位性を確立すること。
(出典：経済産業省, DX推進ガイドライン)

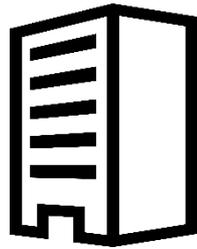
DXの全体像

目的は「競争上の優位性を確保すること」である。

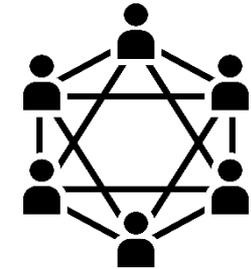
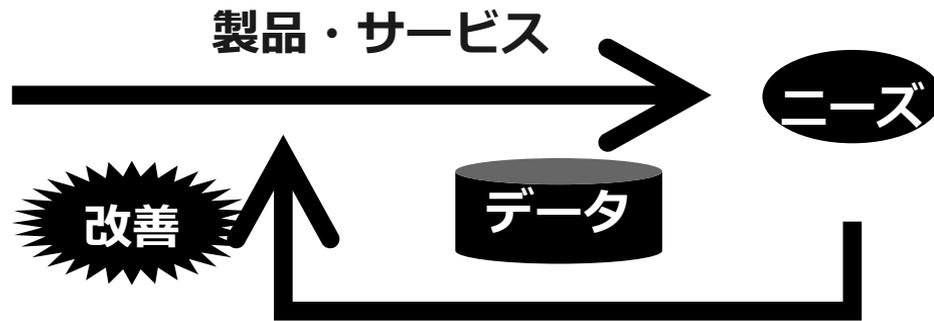
① ビジネスモデル設計

② 製品をデジタルで提供

③ ニーズ計測・データ化



企業



顧客・社会

意思決定・付加価値

⑤ ビジネスモデル変革

④ データ・デジタル技術活用

⑥ 企業変革

DXの全体像

目的は「競争上の優位性を確保すること」である。

① ビジネスモデル設計

② 製品をデジタルで提供

③ ニーズ計測・データ化



⑤ ビジネスモデル変革

④ データ・デジタル技術活用

⑥ 企業変革

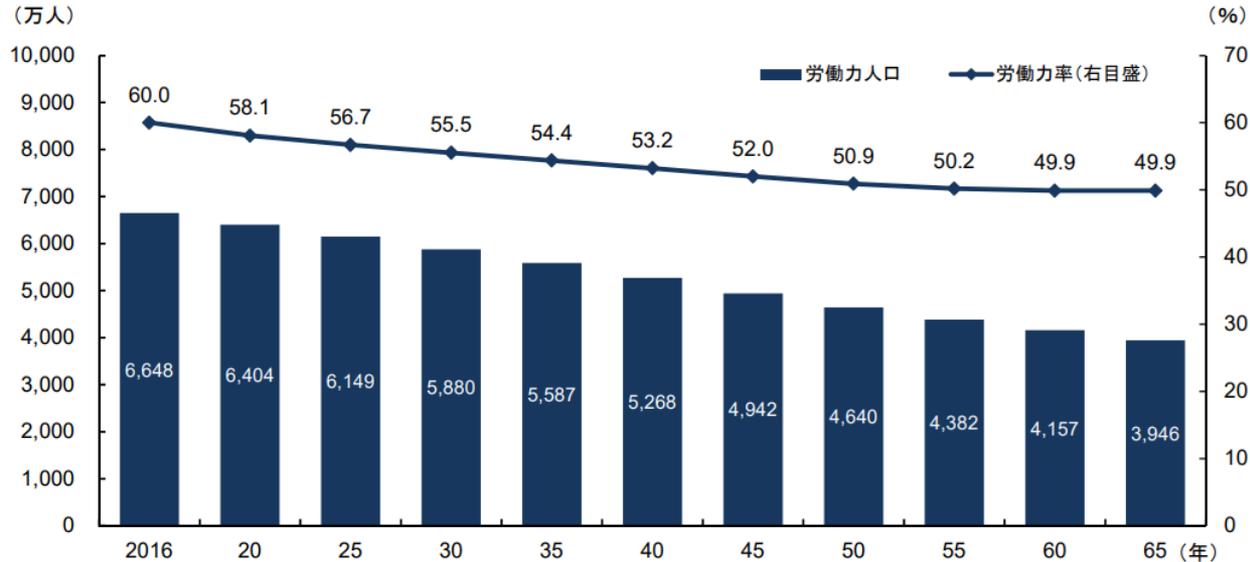
なぜSociety5.0やDXが必要か？

1. 労働力と競争力の低下
2. レガシーシステムの老朽化
3. デジタル技術の波に乗り遅れる

①労働力と競争力の低下

日本はこれから、更なる人手不足が進む。

抜本的な見直しをしない限り、人手不足が事業の継続に直結する課題になり得る。



(注) 2016年は実績。2020年以降は、男女別、年齢5歳階級別の労働力率を2016年と同じとして算出（75歳以上は、2016年の75歳以上の労働力率を75～79歳の労働力率とし、80歳以上はゼロとして算出）。

(資料) 総務省「労働力調査年報」（2016年）、国立社会保障・人口問題研究所「日本の将来推計人口」（2017年4月推計）より、みずほ総合研究所作成

参照:みずほ総合研究所:少子高齢化で労働力人口は4割減

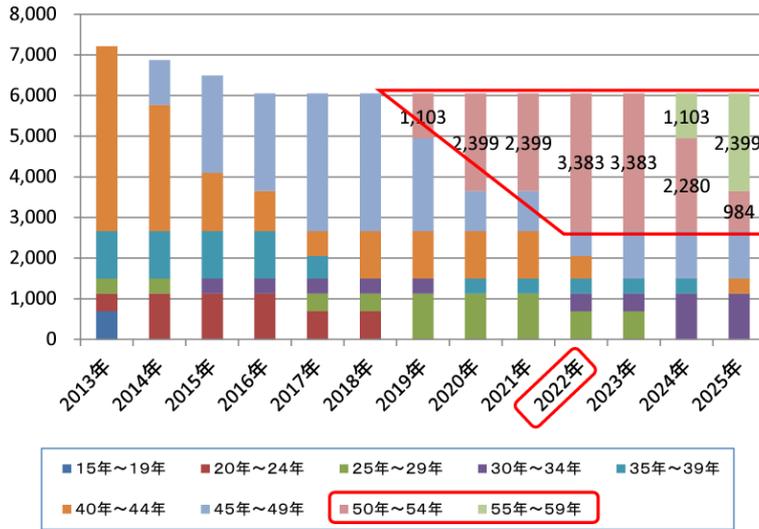
<https://www.mizuho-rt.co.jp/publication/mhri/research/pdf/insight/pl170531.pdf>

②レガシーシステムの老朽化

高度成長期に建設されたプラント等の産業インフラ設備の老朽化が進行し、修繕費も増加。
更に労働人口が減少にもかかわらず、保守・運用のための人材は変わらないorむしろ増加する傾向。

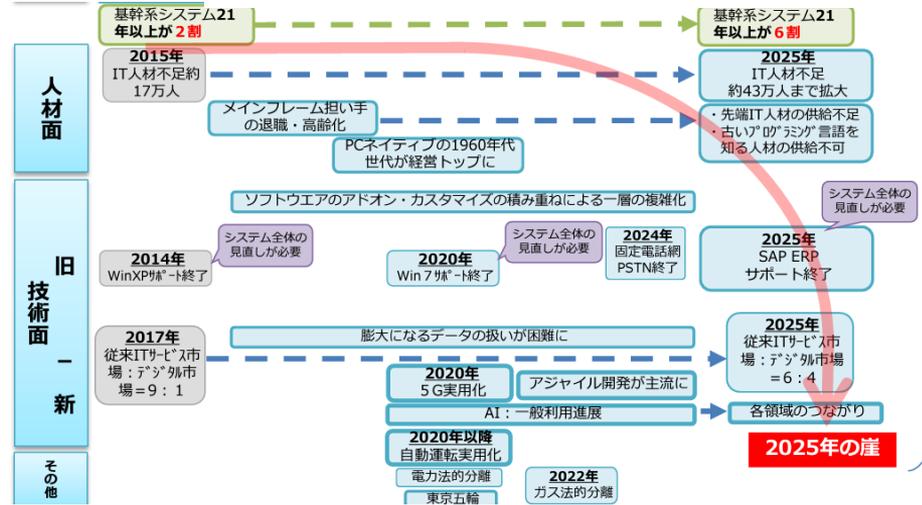
引用：「DXレポート～ITシステム「2025年の崖」の克服とDXの格別な展開～」
https://www.meti.go.jp/shingikai/mono_info_service/digital_transformation/pdf/20180907_01.pdf

○2022年には、半数以上が稼働年数50年以上となる



(資料) 日本の石油化学工業50年データ集 (重化学工業通信社)

引用：「今後の産業保安行政のあり方について」
https://www.meti.go.jp/shingikai/sankoshin/hoan_shohi/pdf/004_03_01.pdf



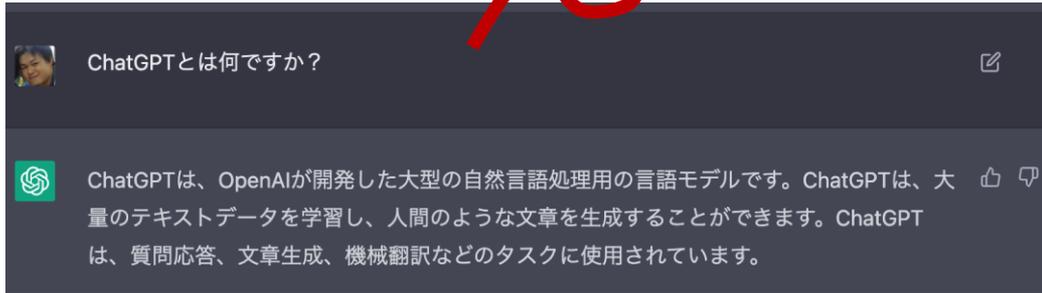
これはDXレポートで言われる「2025年の崖」と同様

③ デジタルの波に乗り遅れる

AIの進化により、状況がどんどん変わり始めている。

プロンプト

ChatGPT



約100億ドル出資



③ デジタルの波に乗り遅れる

生成AIと協働することで、1人で何人分もの価値を生み出す可能性を秘めている

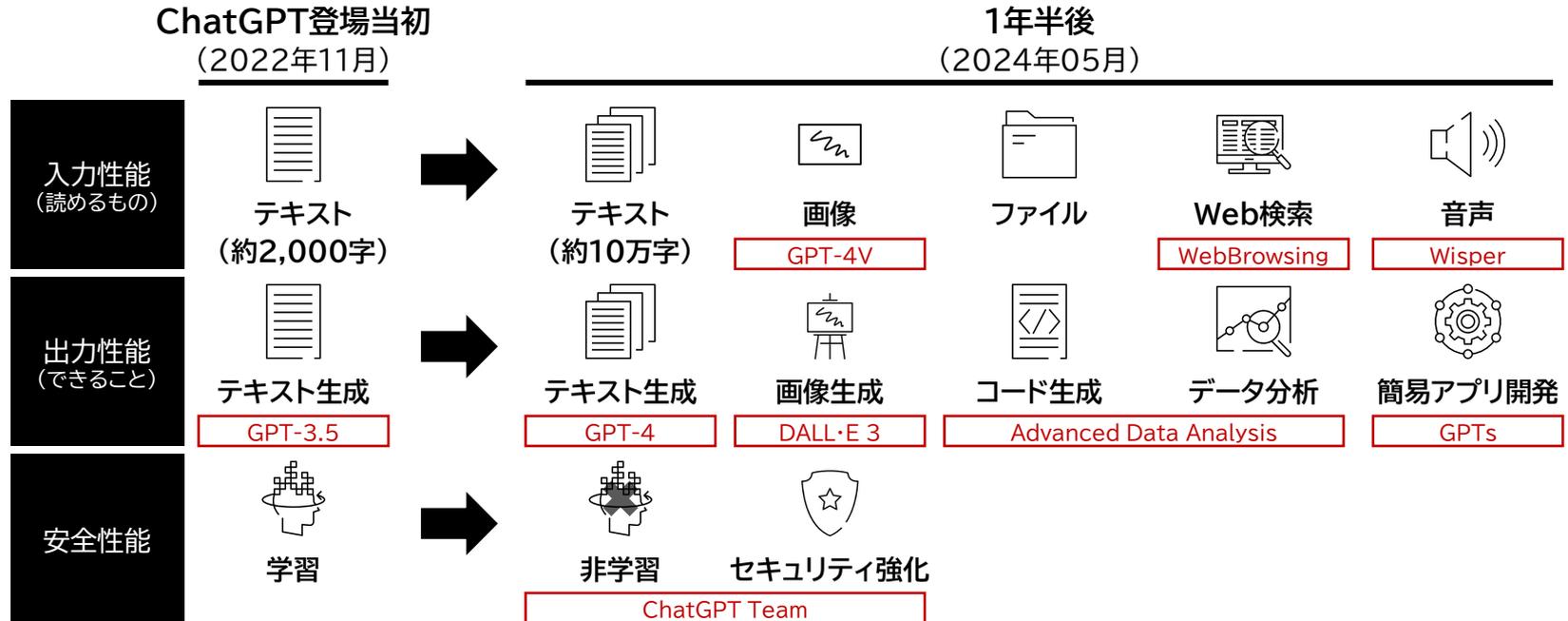
□現在はレベル2→3への過渡期

| レベル | 名称 | 説明 |
|-----|----------------|-----------------|
| 1 | Chatbots | 会話能力を持つ |
| 2 | Reasoners(推論者) | 人間レベルの問題解決能力を持つ |
| 3 | Agents(エージェント) | 自律的に意思決定や行動ができる |
| 4 | Innovators | 発明や発見ができる |
| 5 | Organizations | 組織全体の仕事ができる |

※Bloomberg, (2024/7/11), OpenAI Scale Ranks Progress Toward 'Human-Level' Problem Solving.
<https://www.bloomberg.com/news/articles/2024-07-11/openai-sets-levels-to-track-progress-toward-superintelligent-ai>

③ デジタルの波に乗り遅れる

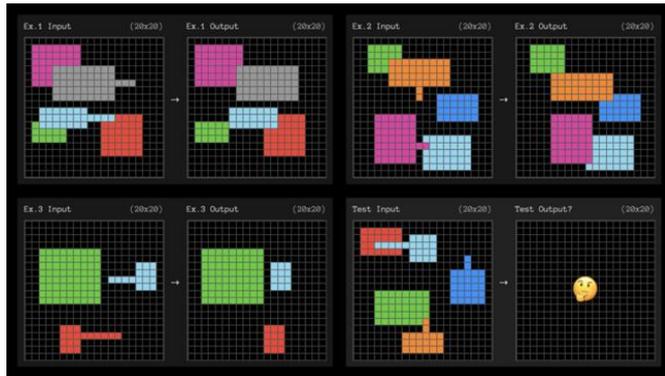
- レベル1:チャットボット(2023-2024)
- 「会話」能力が急速に進化 (マルチモーダル化)



③ デジタルの波に乗り遅れる

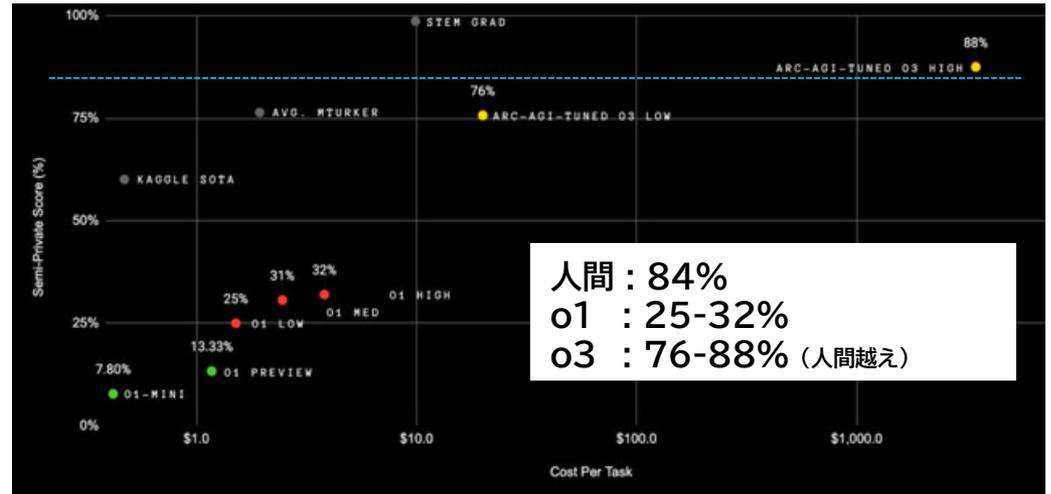
- レベル2:リーズナー/推論者 (2025現在-)
- 生成AIの知能が、いよいよ「人間越え」
 - 推論モデル(o1/o3)の登場。複雑なタスクに対して、多くの時間をかけて推論を行う
 - 科学、数学、コーディングなどで優れた性能、博士のトップレベル水準
 - AIが苦手な抽象度の高い推論でも、人間越え

AGIの性能を評価するベンチマーク
「ARC-AGI」



Source : <https://www.techno-edge.net/article/2024/12/23/3949.html>

抽象度の高い推論でも、人間越えを実現



人間 : 84%
o1 : 25-32%
o3 : 76-88% (人間越え)

③ デジタルの波に乗り遅れる

- レベル3: エージェント
- 自律的×実行を伴う → 複雑な作業を丸っとお任せ！



“1人ユニコーン企業”
たった1人で10億ドルの
価値がある会社を作れる？



③ デジタルの波に乗り遅れる

- レベル3: エージェント
- 自律的×実行を伴う → 複雑な作業を丸っとお任せ！



何もしないとやばそう...



ものづくりの観点からみると？

商品やサービスが顧客に与える価値＝「機能的価値」＋「意味的価値」

「機能的価値」＝品質のよいもの

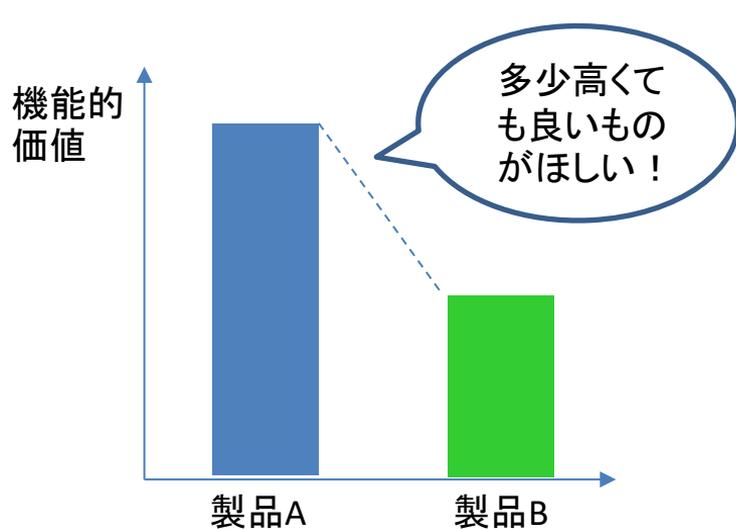
「意味的価値」＝持ってる／使ってることがイケてる！と思える

ものづくりの観点からみると？

商品やサービスが顧客に与える価値＝「機能的価値」＋「意味的価値」

「機能的価値」＝品質のよいもの

「意味的価値」＝持ってる／使ってることがイケてる！と思える



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

6.4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

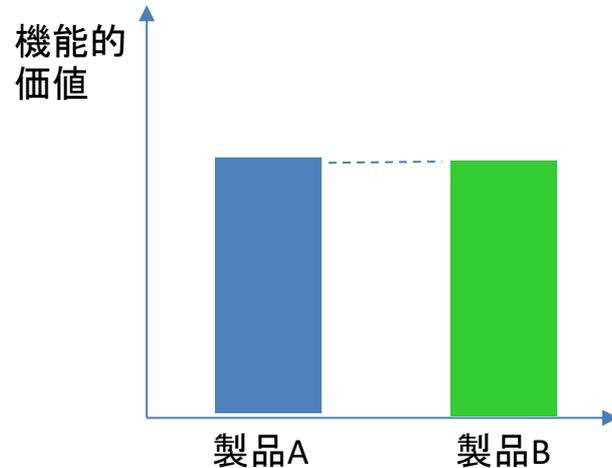
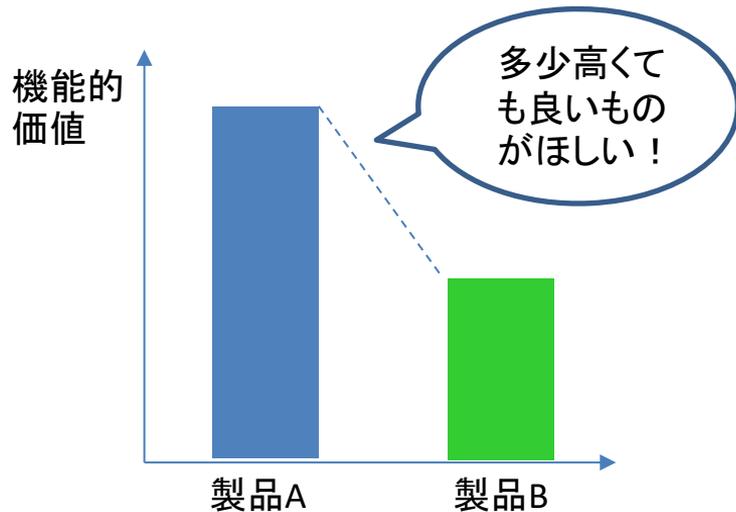
スマホになる
3G→4G、容量大

ものづくりの観点からみると？

商品やサービスが顧客に与える価値＝「機能的価値」＋「意味的価値」

「機能的価値」＝品質のよいもの

「意味的価値」＝持ってる／使ってることがイケてる！と思える

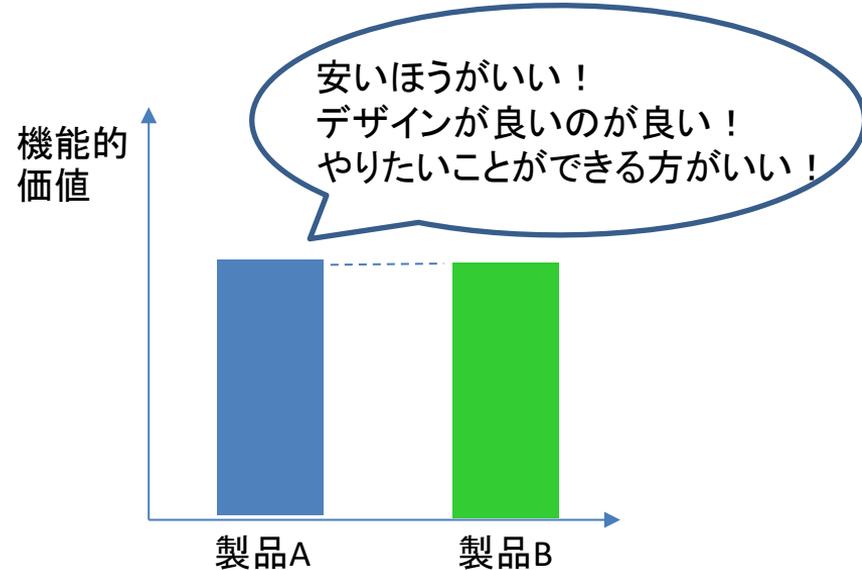
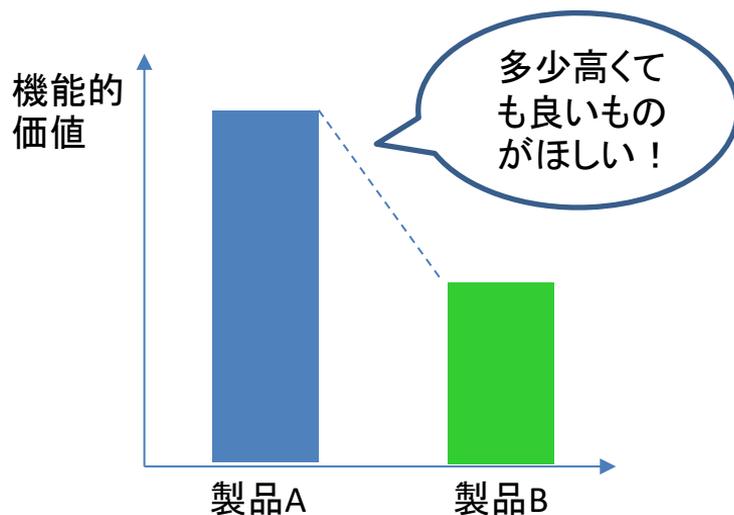


ものづくりの観点からみると？

商品やサービスが顧客に与える価値＝「機能的価値」＋「意味的価値」

「機能的価値」＝品質のよいもの

「意味的価値」＝持ってる／使ってることがイケてる！と思える



ものづくりの観点からみると？

商品やサービスが顧客に与える価値＝「機能的価値」＋「意味的価値」

①安く早く作れる方が強い

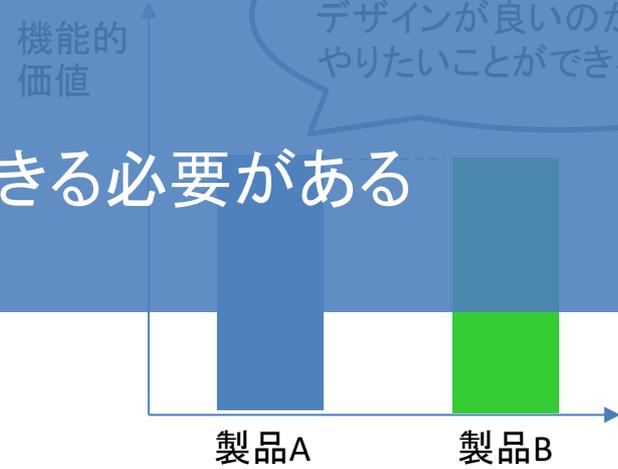
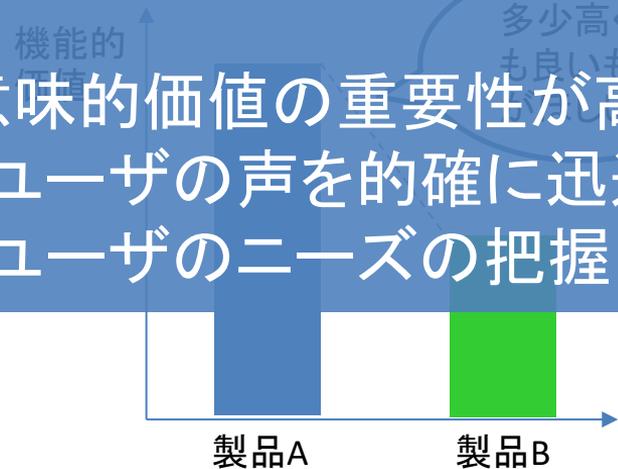
⇒製造工程をより効率化しなければ実現できない

⇒省人化、自動化

②意味的価値の重要性が高くなった

⇒ユーザの声を的確に迅速に対応できる必要がある

⇒ユーザのニーズの把握



ものづくりの観点からみると？

商品やサービスが顧客に与える価値＝「機能的価値」＋「意味的価値」

①安く早く作れる方が強い

⇒製造工程をより効率化しなければ実現できない

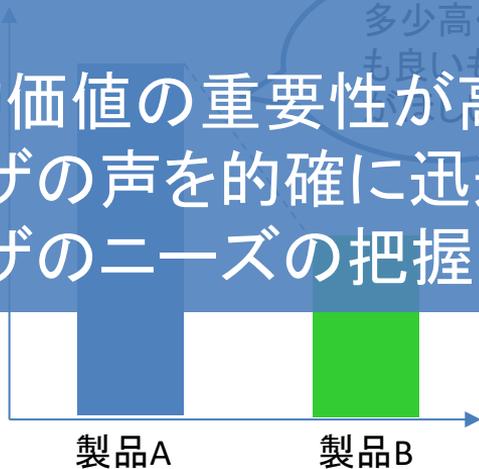
⇒省人化、自動化

②意味的価値の重要性が高くなった

⇒ユーザの声を的確に迅速に対応できる必要がある

⇒ユーザのニーズの把握

機能的
価値



多少高くても良いもの

機能的
価値

AI・データ分析のスキル
が確実に武器になる時代



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

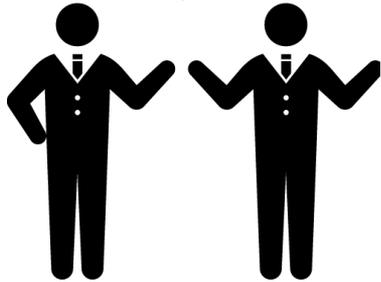
正解はありません



たとえば…

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？



たとえば…

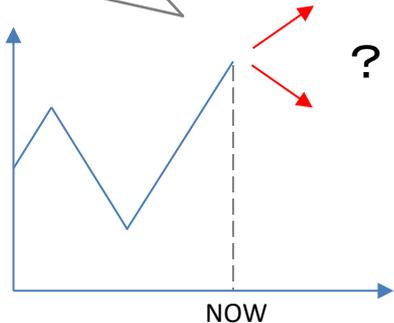
プログラミング の書き方

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

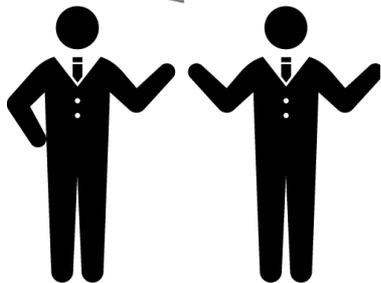
周期的に次は下がる！
いやそのまま上昇する！



たとえば…

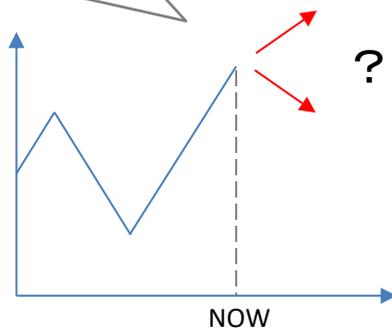
プログラミング の書き方

こっちの方が見やすい！
こっちが動作速くない？



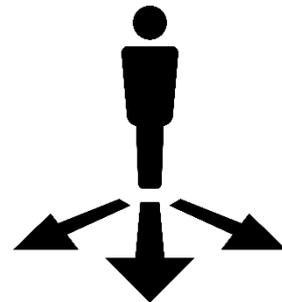
データの解釈

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…

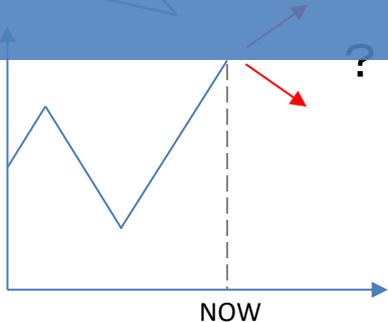
プログラミングの書き方

こっちの方が見やすい！
こっちが動作速くない？



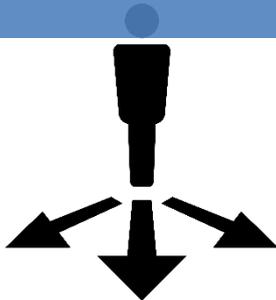
データの解釈

定期的には次は下がる！
いやそのまま上がる！



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…



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

- やって見ないとわからない
- 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

- やって見ないとわからない
- 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

- やって見ないとわからない
- 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

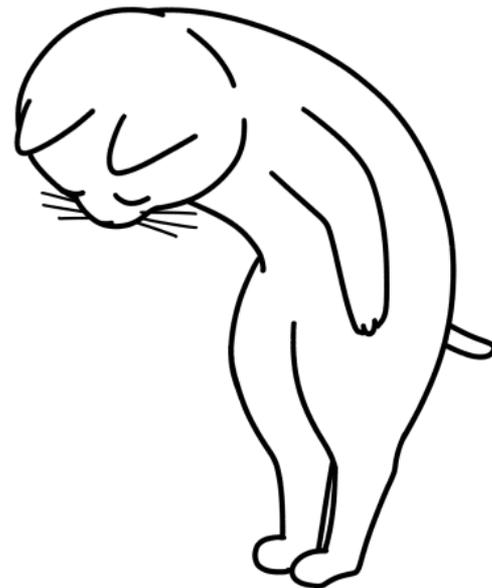
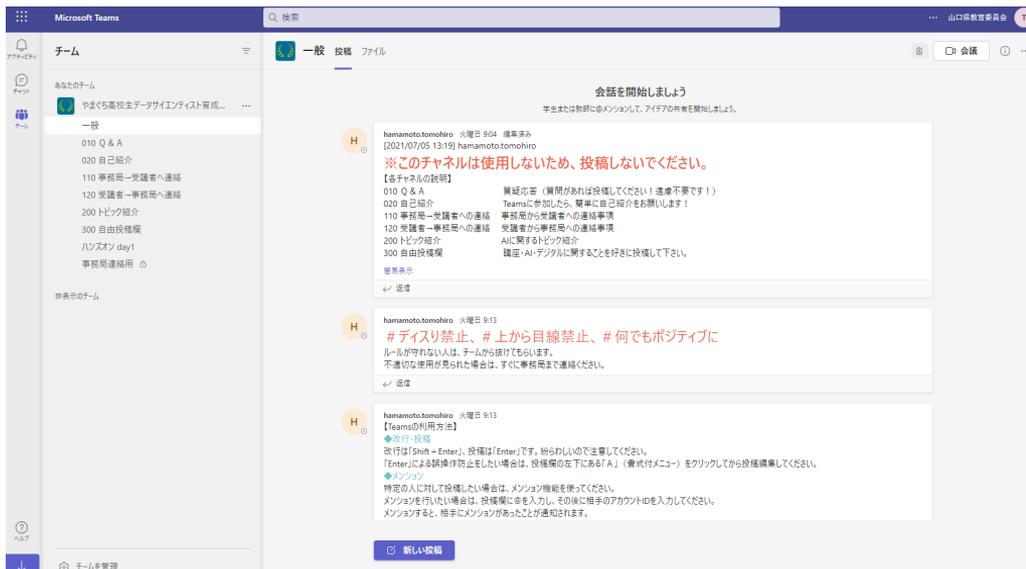
- 考えの整理・洗練化

3. 質問する・教え合う

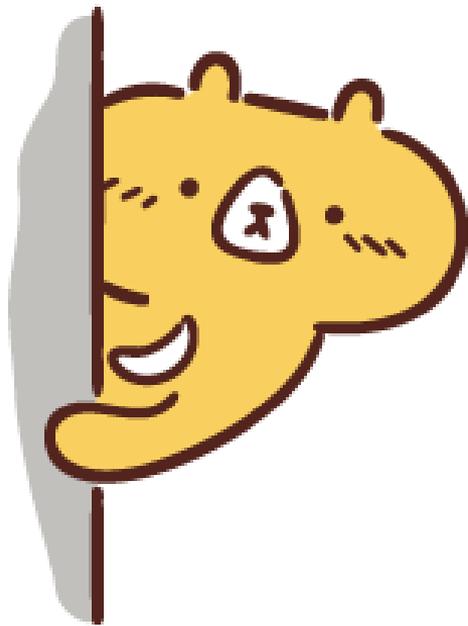
- “わからない”をそのままにしない
- 教えることができるようになる近道



もっと聞きたいこと・知りたいことがあれば teams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



応援しています！

本日のスケジュール

| 時間 | 内容 |
|-------------------|--------------------|
| 10 : 15 ~ 10 : 45 | オリエンテーション |
| 10 : 45 ~ 11 : 15 | データサイエンティストは何者？ |
| 11 : 15 ~ 11 : 25 | 休憩 |
| 11 : 25 ~ 12 : 00 | teamsに触ってみよう |
| 12 : 00 ~ 13 : 00 | 休憩 |
| 13 : 00 ~ 14 : 00 | Pythonに触ってみよう①（座学） |
| 14 : 00 ~ 14 : 10 | 休憩 |
| 14 : 10 ~ 15 : 30 | Pythonに触ってみよう② |
| 15 : 30 ~ 15 : 40 | 休憩 |
| 15 : 40 ~ 16 : 50 | Pythonに触ってみよう③ |
| 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、

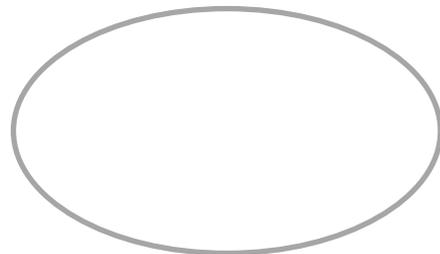
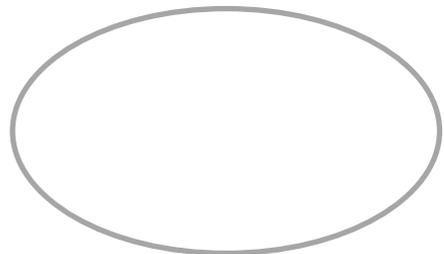
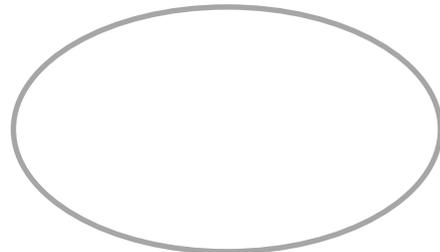
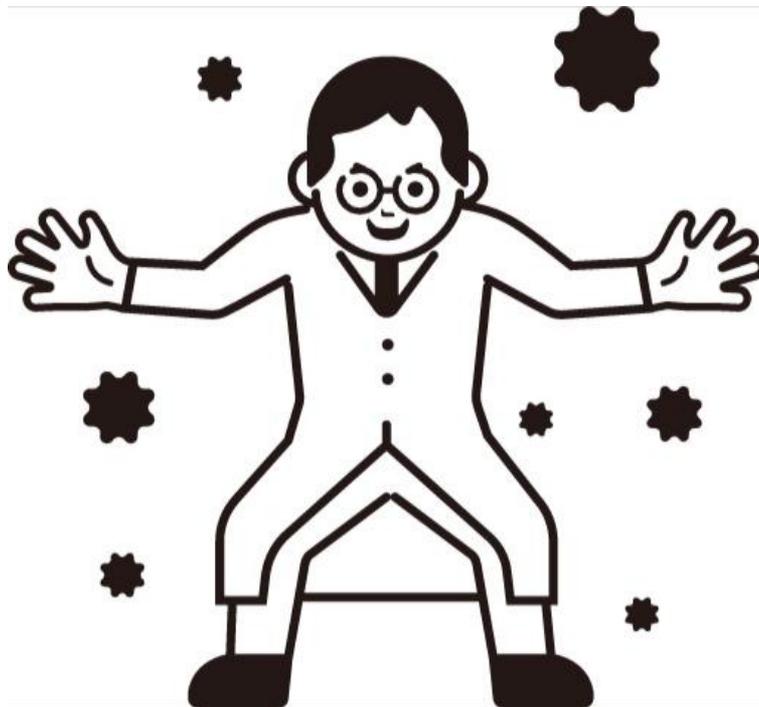
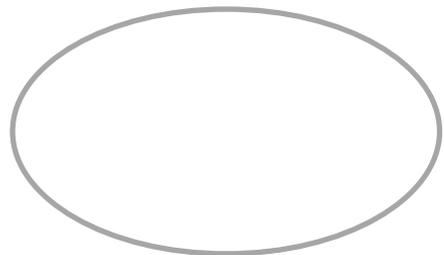
データサイエンティスト

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

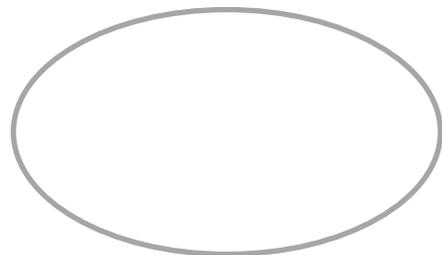
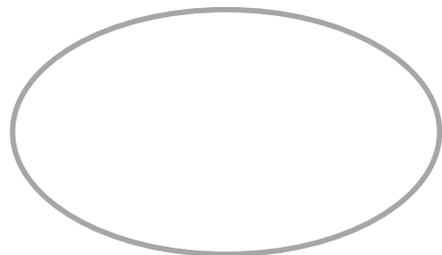
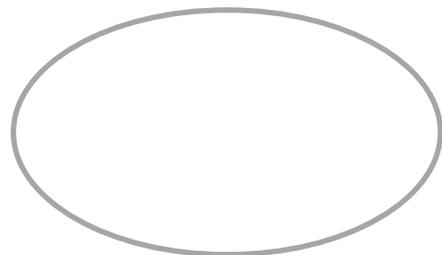
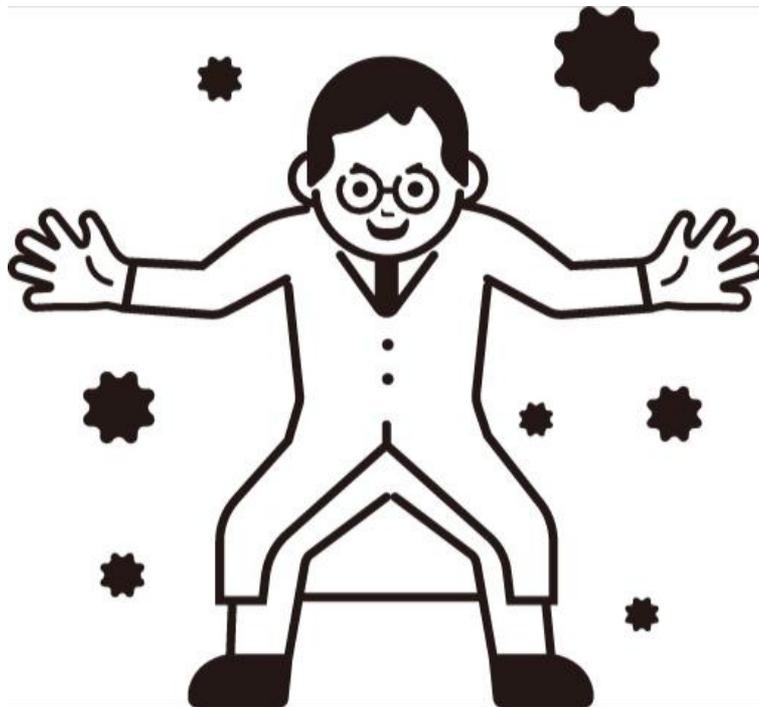
を目指してもらいたい、応援したい！

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？

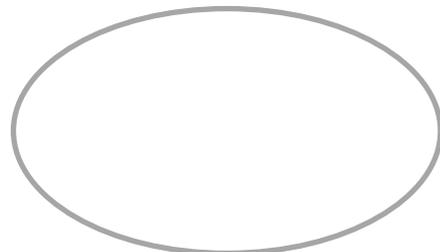
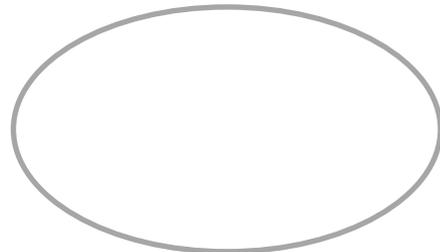
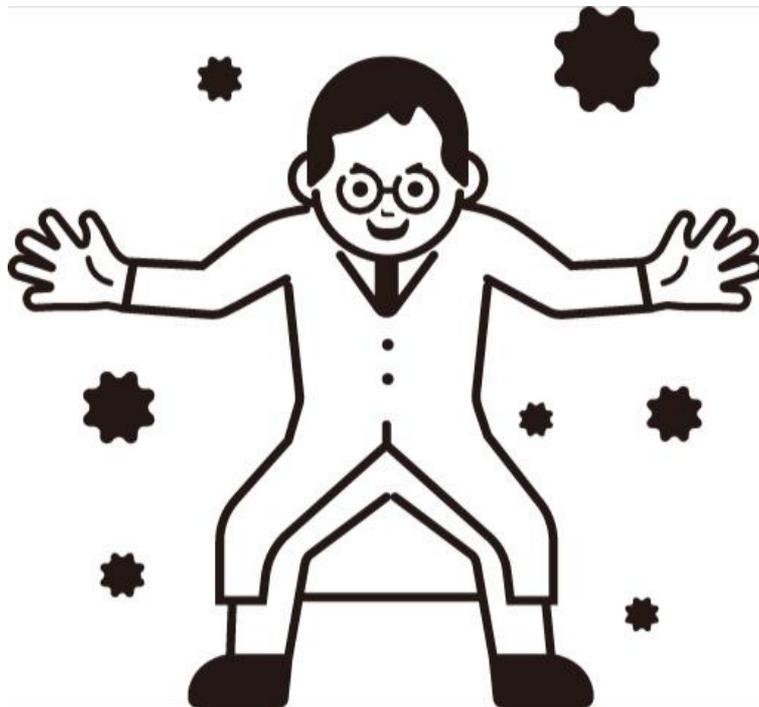
数学・統計



データサイエンティストってどんなイメージ？

数学・統計

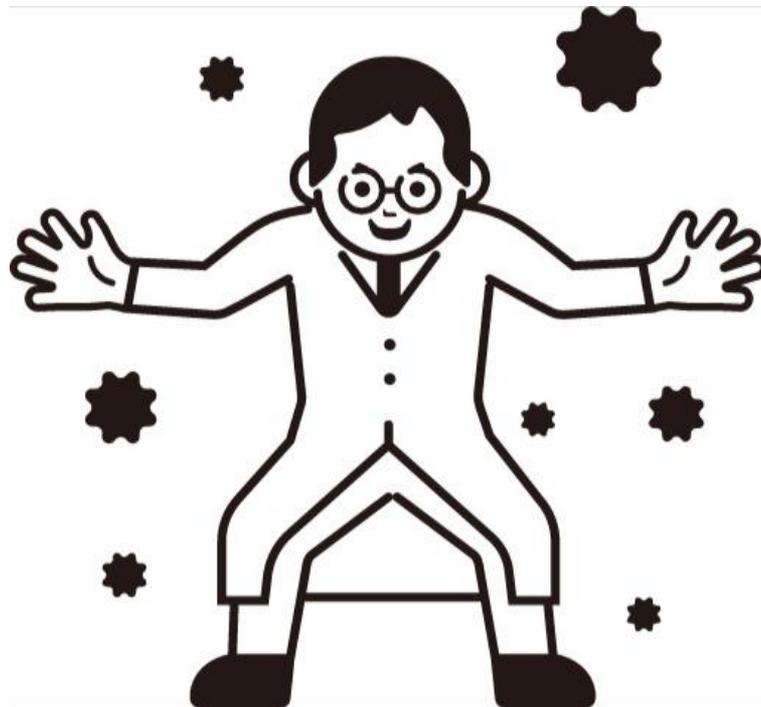
プログラム



データサイエンティストってどんなイメージ？

数学・統計

プログラム

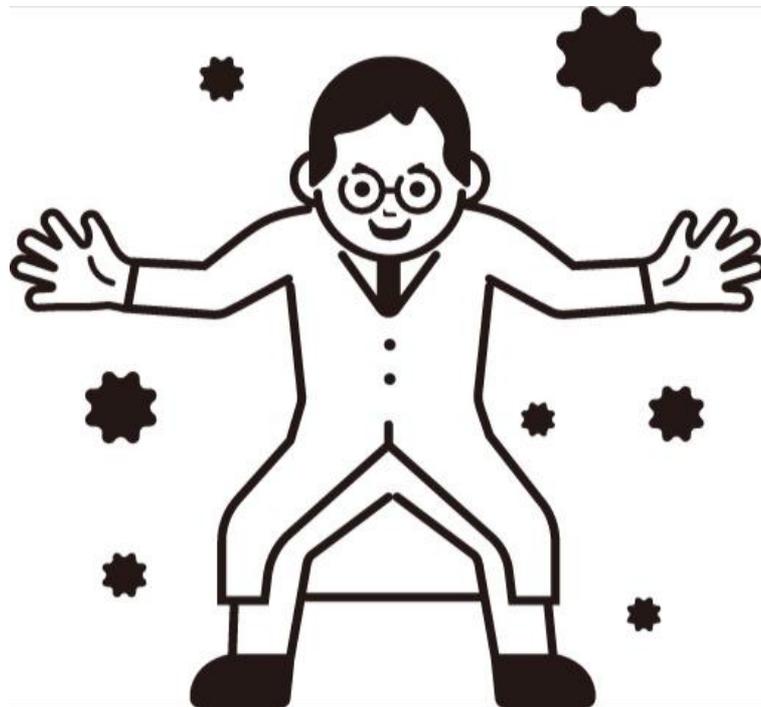


データ分析

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



データ分析

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

ビジネスカ

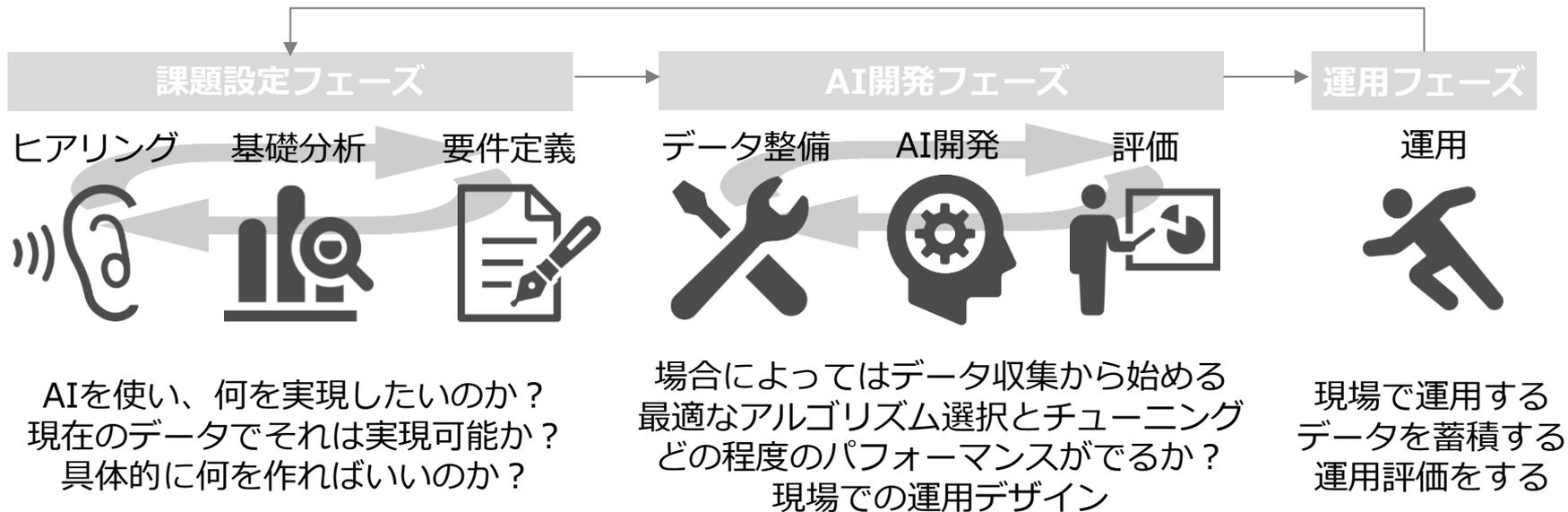
データサイエンスを意味のある形に使えるようにし、実装、運用できるようにする力

データ
サイエンスカ

データ
エンジニアリングカ

データサイエンティストの仕事内容

▼作業内容（例）



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きるか？
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
- ②sample.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

```
print('Hello World!')
```

Python (①)

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

Python (②)

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

Java (①)

```
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

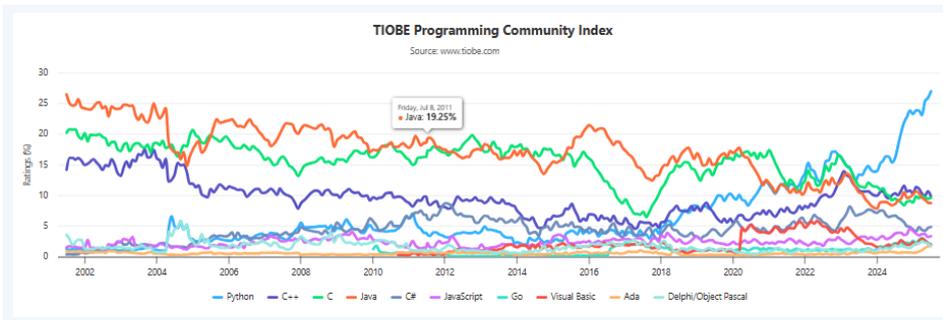
```
model = LR()  
model.fit(X.reshape(-1,1),y)
```

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
- そこから現在に至るまでずっと1位をキープ
- 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** (!)

| Jul 2025 | Jul 2024 | Change | Programming Language | Ratings | Change |
|----------|----------|--------|--|---------|---------|
| 1 | 1 | |  Python | 26.98% | +10.85% |
| 2 | 2 | |  C++ | 9.80% | -0.53% |
| 3 | 3 | |  C | 9.65% | +0.16% |
| 4 | 4 | |  Java | 8.76% | +0.17% |
| 5 | 5 | |  C# | 4.87% | -1.85% |
| 6 | 6 | |  JavaScript | 3.36% | -0.43% |
| 7 | 7 | |  Go | 2.04% | -0.14% |
| 8 | 8 | |  Visual Basic | 1.94% | -0.13% |
| 9 | 24 | ↗ |  Ada | 1.77% | +0.99% |
| 10 | 11 | ↗ |  Delphi/Object Pascal | 1.77% | -0.12% |



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

おまけ：バイブコーディング (vibe coding)

AIに依存したプログラミング手法で、人は解きたい問題を、コーディングに特化した大規模言語モデル(LLM)へのプロンプトとして自然言語で記述

python+seleniumを使った自動化プログラムを作成します。
理解できましたら、はいと回答してください。

2～3秒の間、理由付けた自動化プログラムの作成
はい

以下のhtmlに対し、メールアドレスとパスワードを入力してサインインするプログラムを書いてください。

```
# html
<!DOCTYPE html>
<html la
```

以下は Python の Selenium を使用して、指定の HTML のサインインフォームに対してメールアドレスとパスワードを入力し、サインインする自動化プログラムのサンプルコードです。
なお、コード中の URL や入力するメールアドレス、パスワードは適宜ご自身の環境に合わせて変更してください。

```
python
from selenium import webdriver
```

```
save_button = wait.until(
    EC.element_to_be_clickable((By.XPATH, "//input[@type='submit' and @value='保存する']")))
save_button.click()
# 保存後の画面遷移を確認するため、必要に応じて待機
time.sleep(3)

In [60]: url = "https://ms.quest.signate.jp/sign_in"
email = "t.takada@signate.co.jp"
password = "cogitoergosum"

path = "../../../Downloads/問題データセット_ブロンズ_02_why_.json"
absPath = os.path.abspath(path)
print(absPath)

driver = webdriver.Chrome(options=options)
driver.get(url)

sign_in(email, password, driver)
click_question_dataset(driver)
click_new_creator(driver)
select_skill_and_save(driver)
go_to_last_page(driver)
click_last_dataset(driver)
click_new_question(driver)
click_bulk_register(driver)
bulk_upload_json(driver, absPath)
click_edit_button(driver)
set_status_approved_and_save(driver)

print('Completed')

driver.quit()

/Users/t.takada/Downloads/問題データセット_ブロンズ_02_why_.json
Completed

In [ ]:
In [ ]:
```

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

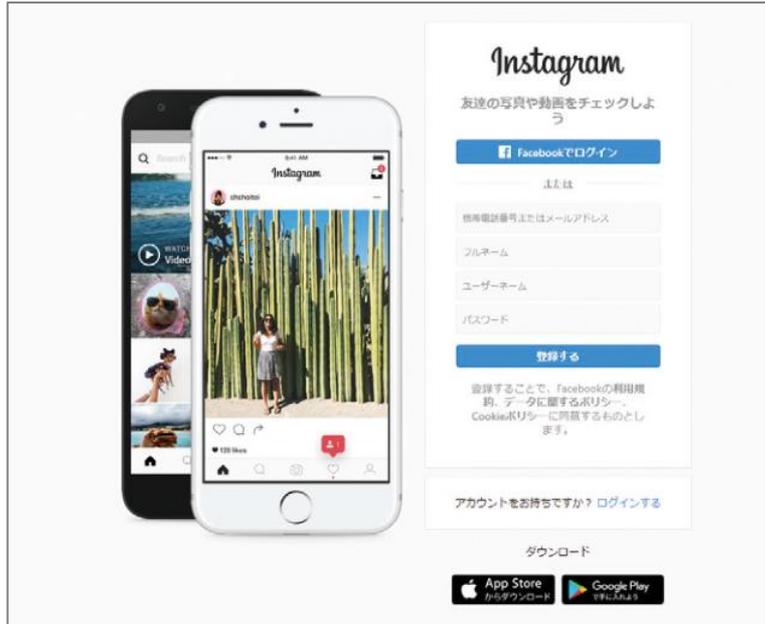
Pythonでできること

- 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

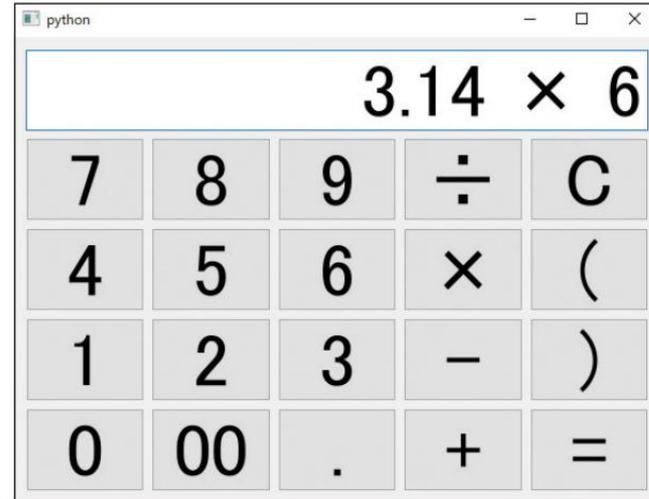
Pythonでできること

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

| 項目 | 数値目標 | 実績 | 差異 | 達成率 |
|-----|------|------|----|------|
| 売上 | 3000 | 3000 | 0 | 100% |
| 高橋 | 2600 | 2600 | 0 | 100% |
| 金野 | 400 | 400 | 0 | 100% |
| 達成率 | 87% | 87% | -5 | 100% |
| 経費 | 100 | 100 | 0 | 100% |
| 高橋 | 10 | 10 | 0 | 100% |
| 相沢 | 100 | 100 | 0 | 100% |
| 達成率 | 5 | 5 | 0 | 100% |
| 経費 | 100 | 100 | 0 | 100% |
| 高橋 | 100 | 100 | 0 | 100% |
| 相沢 | 100 | 100 | 0 | 100% |
| 達成率 | 100% | 100% | 0 | 100% |

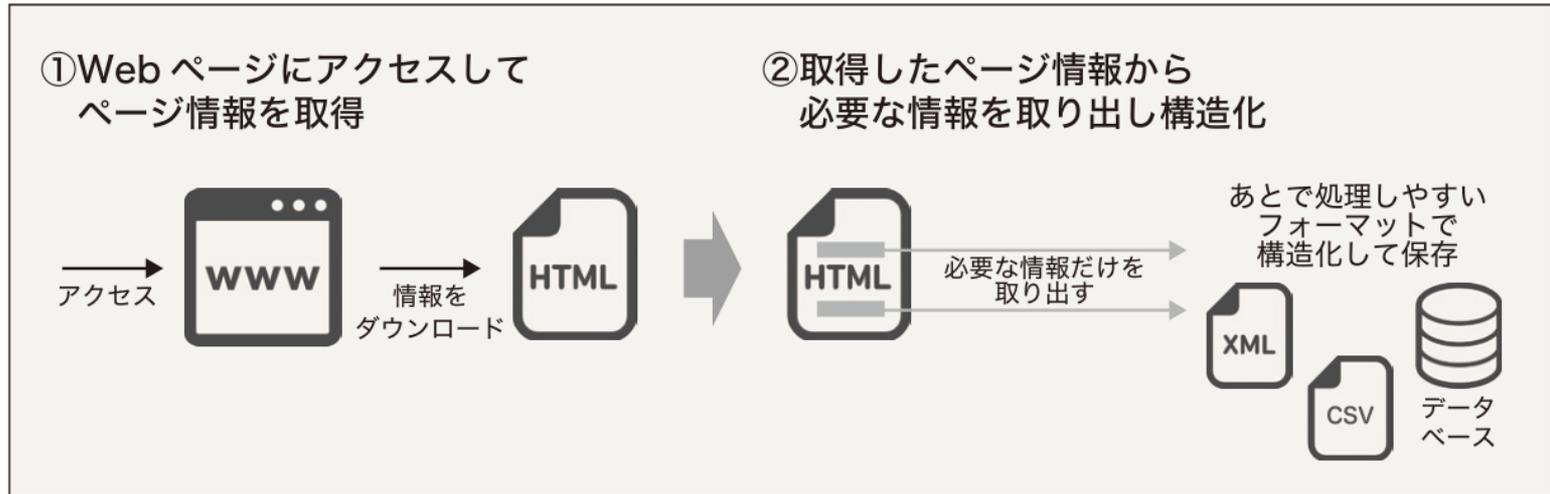
バラバラに散らばったExcelファイルを
まとめて一覧化・集計を行う など

| 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 | 小林〇〇 | 3000 | 2600 | -400 |
| 3 | 2020年01月31日 | A支店 | 新谷〇〇 | 3300 | 3250 | -50 |
| 4 | 2020年01月31日 | A支店 | 竹田〇〇 | 3000 | 3100 | 100 |
| 5 | 2020年01月29日 | B支店 | 平山〇〇 | 2200 | 2100 | -100 |
| 6 | 2020年01月31日 | B支店 | 金城〇〇 | 2200 | 1980 | -220 |
| 7 | 2020年01月31日 | C支店 | 丸高〇〇 | 3100 | 3700 | 600 |
| 8 | 2020年01月30日 | C支店 | 小田島〇〇 | 3100 | 3500 | 400 |
| 9 | 2020年01月31日 | C支店 | 青木〇〇 | 3100 | 3350 | 250 |
| 10 | 2020年01月31日 | D支店 | 佐々木〇〇 | 2500 | 2600 | 100 |
| 11 | 2020年01月31日 | D支店 | 島村〇〇 | 2500 | 2800 | 300 |
| 12 | 2020年01月31日 | E支店 | 畑中〇〇 | 1900 | 1920 | 20 |
| 13 | 2020年01月31日 | E支店 | 米沢〇〇 | 1900 | 1700 | -200 |
| 14 | | | | | | |
| 15 | | | | | | |
| 16 | | | | | | |

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

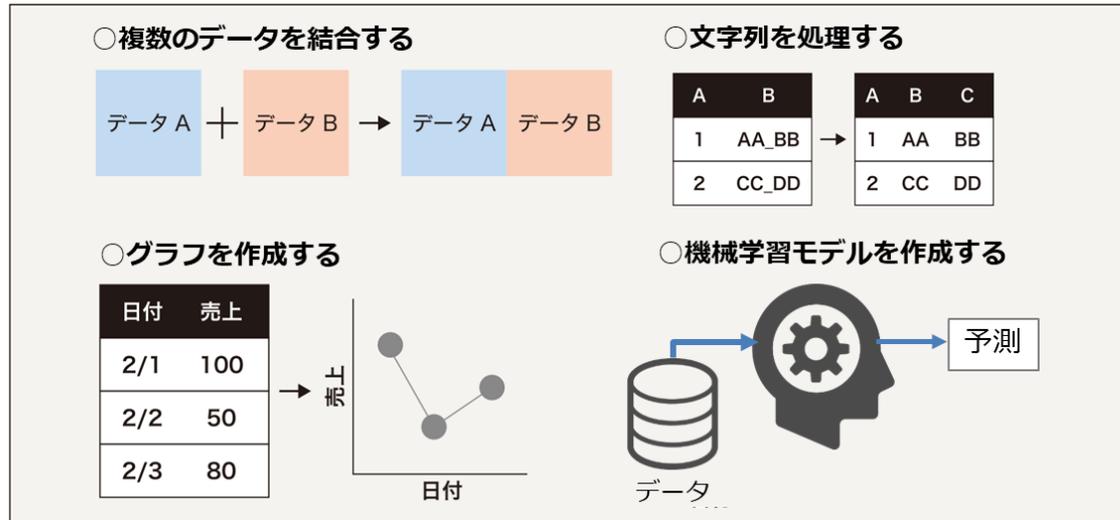


※ただし、スクレイピングを禁止しているWebサイトもあるので注意が必要です

Pythonでできること

データ分析・機械学習

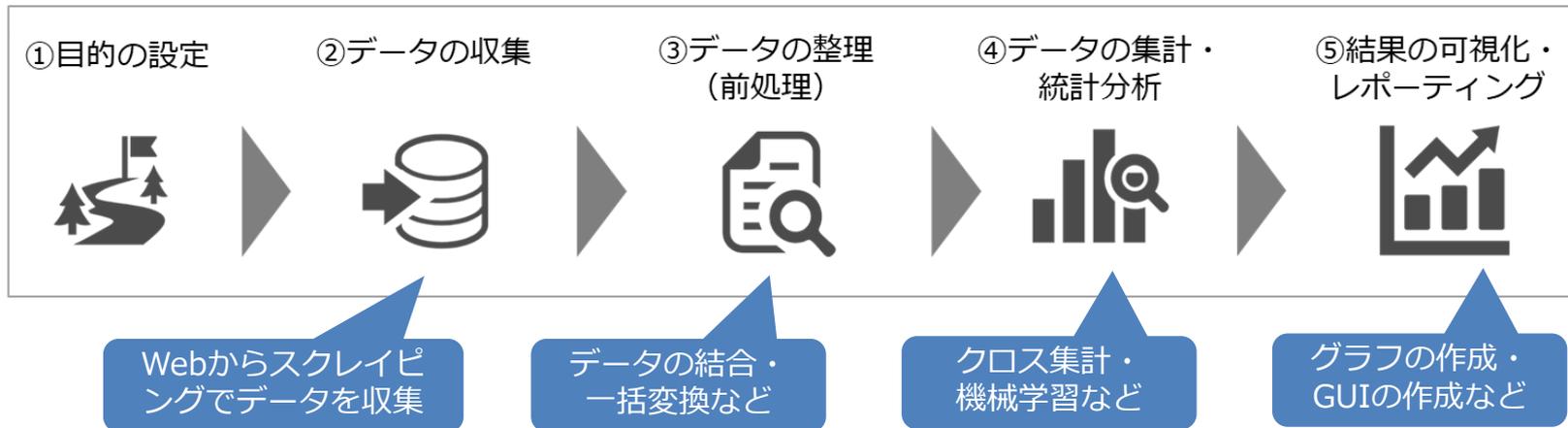
- データ分析では、グラフの作成や機械学習モデルの作成を行うだけでなく、事前にデータを分析しやすいように加工する「前処理」という作業も必要になる
- 例えば、複数のデータを1つに結合する、一括でデータの値を変換する など



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>

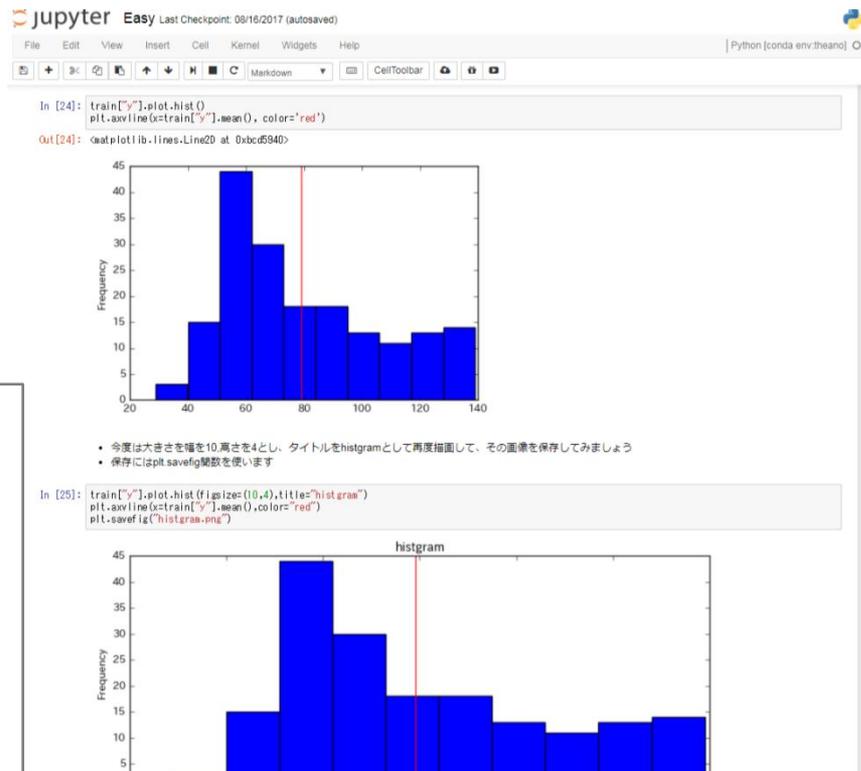
- Pythonの基礎（文法的なお話）
- SIGNATE Questを活用してPythonや機械学習を学ぶ方法の手引き

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



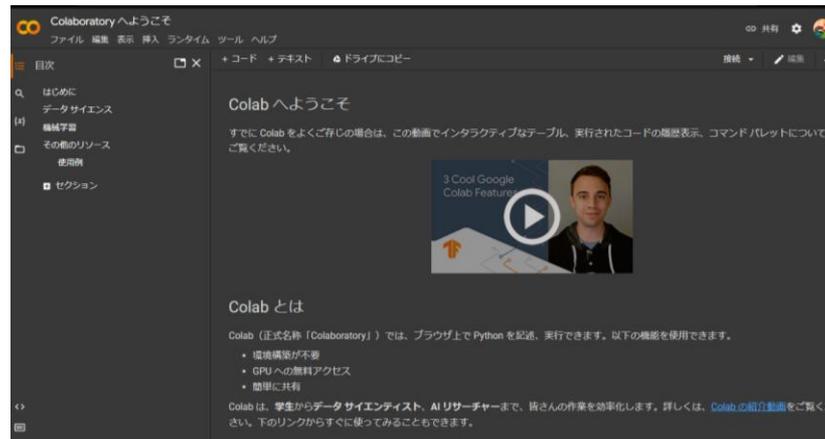
Google Colaboratoryとは？

Google Colaboratory



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

- **Teams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://cloud.signate.jp/sign_in

- **Google Colabを使います**

- <https://colab.research.google.com/?hl=ja>

Teamsの使い方

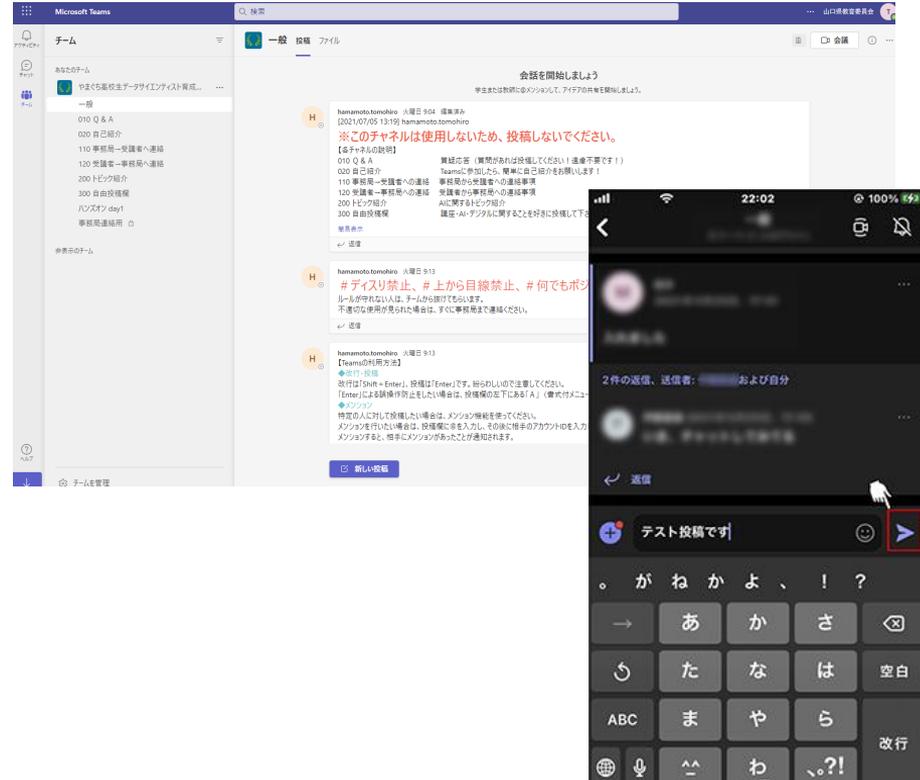


Teamsとは？

What is Teams ?

Microsoftが提供するコミュニケーションアプリです。

Teams を使うことで、リアルタイムなコミュニケーション、会議、共同作業、ファイル共有などが他メンバーと手軽にできます。



Teamsとは？

チーム

The screenshot shows the Microsoft Teams interface. On the left, the 'チーム' (Team) sidebar is visible, listing various channels. A callout box labeled 'チーム' points to the sidebar. The main area shows a channel named 'やまぐち高校生データサイエンティスト育成...'. A callout box labeled 'チャンネル' points to this channel name. Below the channel name, a list of channels is shown, including '一般', '010 Q & A', '020 自己紹介', '110 事務局→受講者へ連絡', '120 受講者→事務局へ連絡', '200 トピック紹介', '300 自由投稿欄', 'ハズオン day1', and '事務局連絡用'. A callout box labeled '投稿メッセージ' points to a message in the chat area. The message content includes: 'hamamoto.tomohiro 火曜日 9:04 編集済み [2021/07/05 13:19] hamamoto.tomohiro ※このチャンネルは使用しないため、投稿しないでください。【各チャンネルの説明】 010 Q & A 質疑応答（質問があれば投稿してください！遠慮不要です！） 020 自己紹介 Teamsに参加したら、簡単に自己紹介をお願いします！ 110 事務局→受講者への連絡 事務局から受講者への連絡事項 120 受講者→事務局への連絡 受講者から事務局への連絡事項 200 トピック紹介 AIに関するトピック紹介 300 自由投稿欄 講座・AI・デジタルに関することを好きに投稿して下さい。 非表示 返信' and '# ディスリ禁止、# 上から目線禁止、# 何でもポジティブに ルールが守れない人は、チームから抜けてもらいます。 不適切な使用が見られた場合は、すぐに事務局まで連絡ください。 返信'. Below this, another message says 'hamamoto.tomohiro 火曜日 9:13 【Teamsの利用方法】 ◆改行・投稿 改行は「Shift + Enter」、投稿は「Enter」です。紛らわしいので注意してください。 「Enter」による誤操作防止をしたい場合は、投稿欄の左下にある「A」（書式付メニュー）をクリックしてから投稿編集してください。 ◆メンション 特定の人に対してメンションを行いたい場合は、投稿欄の左下にある「A」（書式付メニュー）をクリックしてから投稿編集してください。 返信'. A callout box labeled '新しい投稿' points to the '新しい投稿' button at the bottom of the chat area.

投稿メッセージ

新しい投稿

Teamsとは？

会話を開始しましょう
学生または教師に@メンションして、アイデアの共有を開始しましょう。

H hamamoto.tomohiro 火曜日 9:04 編集済み
[2021/07/05 13:19] hamamoto.tomohiro
※このチャンネルは使用しないため、投稿しないでください。
【各チャンネルの説明】
010 O & A 質疑応答（質問があれば投稿してください！遠慮不要です！）
詳細表示

返信

H hamamoto.tomohiro 火曜日 9:13
#ディスリ禁止、#上から目線禁止、#何でもポジティブに
ルールが守れない人は、チームから抜けてもらいます。
不適切な使用が見られた場合は、すぐに事務局まで連絡ください。
← 返信

H hamamoto.tomohiro 火曜日 9:13
【Teamsの利用方法】
◆改行・投稿
改行は「Shift + Enter」、投稿は「Enter」です。紛らわしいので注意してください。
「Enter」による誤操作防止をしたい場合は、投稿欄の左下にある「A」（書式付メニュー）をクリックしてから投稿編集してください。
詳細表示

新しい投稿

1つの話題に返信をすることができる

※チャットは情報がすぐ流れてしまうため、1つの話題に対して議論を深めたい場合には返信を使って会話するのがベターです

Teamsとは？

スタンプ

コミュニケーションツール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。

いろいろ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の例

スタンプ

を開始しましょう。



い。

【です！）

イブに

をクリックしてから投稿編集してください。

楽しいコンテンツを検索してください

スマイル

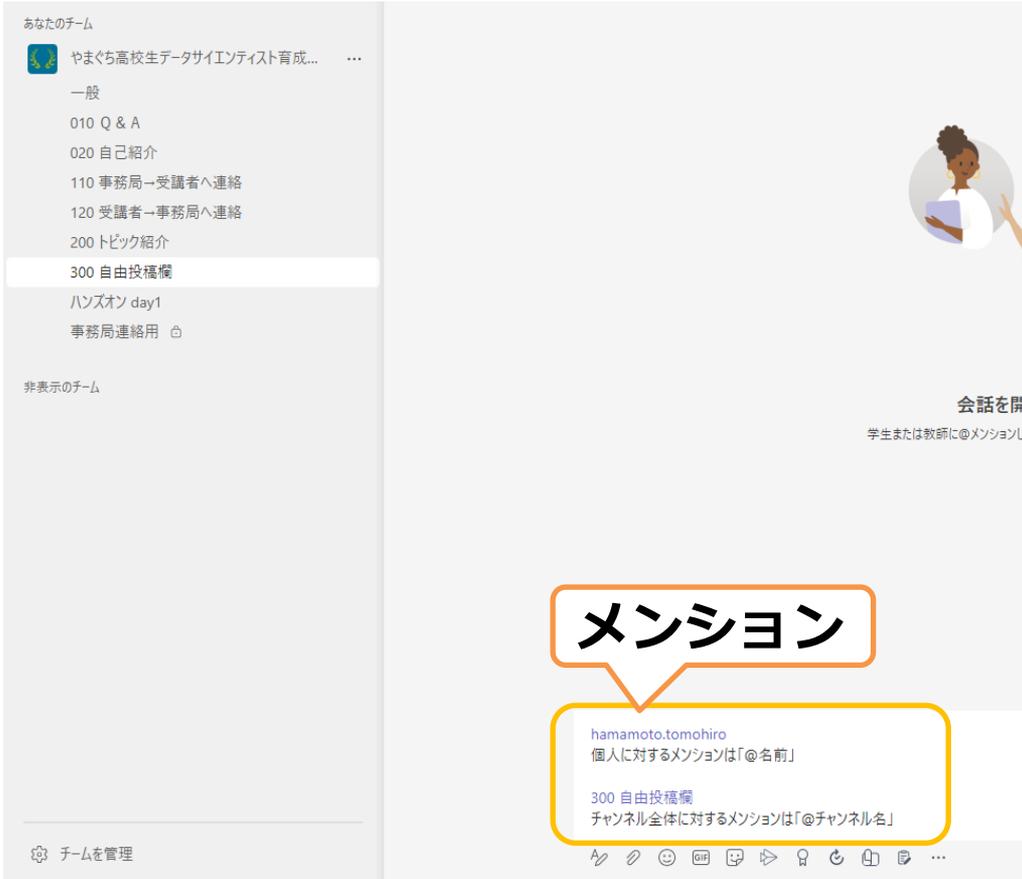


Teamsとは？

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。



@名前

→名前の人に通知

@channel

→チャンネルの全員に通知

Teams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* |
| <i>Italicize</i> | 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>Code</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>Code block</code> | 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、 <code>スペース</code> キーを押します： 1. 対象のテキスト |
| • Bulleted list | 行をインデントするには、 <code>Tab</code> を押します。 テキストの前にアスタリスクを追加して、 <code>スペース</code> 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、 <code>Tab</code> を押します。 |

よく使うもの

```はshift+@

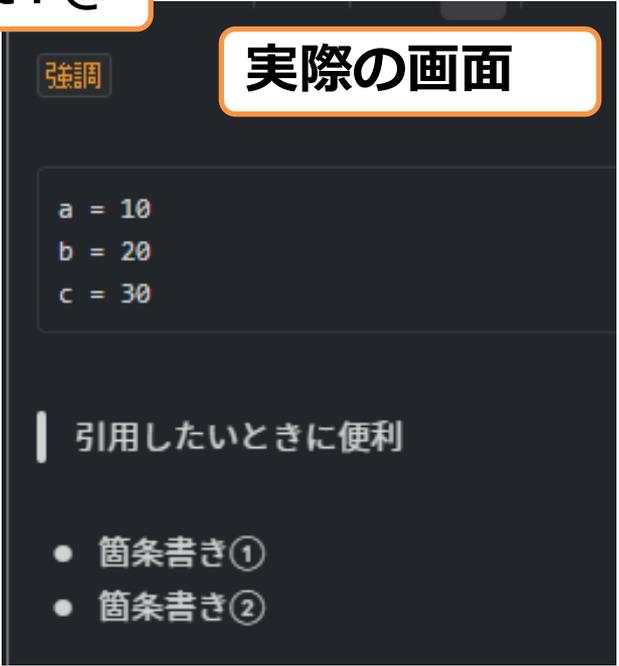
``強調``

`````  
`コードブロック`
`````

`>`引用

- 箇条書き①  
- 箇条書き②

実際の画面



-と箇の間にスペースが必要

# Teams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

- ① 「300 自由投稿欄」チャンネルを開く
- ② 以下の投稿を行ってください。
  - ・ とりあえず何か挨拶を投稿する
  - ・ 文章の中に「ボールド (\*aa\*) 」や「イタリック (\_aa\_) 」を含んだ投稿をする
  - ・ 文章の中に「強調表現 (`aa`）」を含んだ投稿をする（`はshift+@)
  - ・ 文章の中に「引用 (>）」を含んだ投稿をする
  - ・ 文章の中に「コードブロック (```) で囲む」を含んだ投稿をする
- ③ 上記が出来たら次のページの課題に進む

# これからやっていただきたいこと②

- ① 「020 自己紹介」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／最近ハマっていること：
  - ・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、返信機能を使い、投稿時には返信元の名前に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、返信機能で回答する
- ⑥ 回答に対し、お互いにスタンプやメッセージで反応する

# Jupyter Notebookの使い方



# Jupyter Notebook

NotebookのUIは次のようになっています。



# Challenge MissionのUI

The screenshot displays the Challenge Mission interface, which is divided into two main sections: a left-hand sidebar for instructions and a right-hand main area for code execution.

**Left Sidebar (Instructions):**

- 説明 (Description):** ファイル一覧 (File List) | データ内容 (Data Content)
- 優良顧客を探せ！～顧客ターゲティング～**
- Challenge Missionでは、Questで取り組んだ定期預金キャンペーンの申込率予測にゼロからチャレンジします。**
- データはQuestで使用したものと同一データです。**
- また、機械学習モデルはQuestと同様、決定木を使用し、random\_stateは0としましょう。**

---

**データは、学習用データの train.csv、評価用データの test.csv および、予測結果を提出する際に必要となる sample\_submission.csv が用意されています。**

**データの中身の詳細は、データ内容タブに記載しています。**

**なおデータはディレクトリ ../input 以下にあります。例えば学習用データ train.csv を読み込む場合、以下のように記述します。**

```
pd.read_csv("../input/train.csv")
```

---

**予測ができれば、採点するための投稿用ファイルを作成します。**

**サンプルファイルが用意されていますので、まずサンプルファイルを読み込みます。**

```
submit = pd.read_csv("../input/sample_submission.csv", header=None)
```

**サンプルファイルはヘッダーがないため、引数 header に None を指定しています。**

**ファイルの内容としては、0列目にtestID、1列目にダミーデータが入っています。この1列目の値には、作成した予測値が入ります。**

**Right Main Area (Code Editor):**

- Toolbar:** Save, + Add, % Cut, ↑ Up, ↓ Down, ▶ Run, ⏻ Run All, ■ Stop, Code (dropdown), 🔄 Restart Kernel
- Status:** (unsaved changes)
- Code Input:** In [ ]:

**Bottom Navigation:** notebook | 投稿履歴 | 投稿する

# 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。

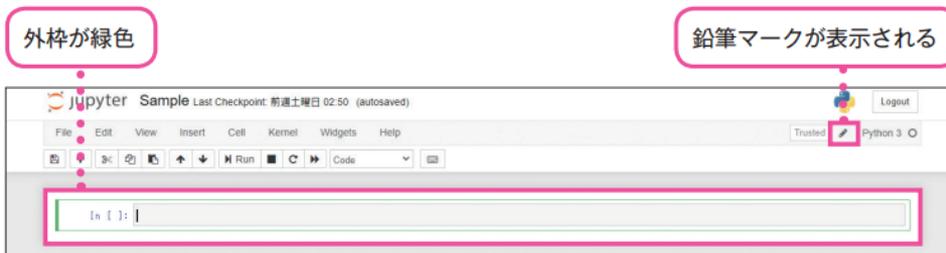
「コマンドモード」・・・セルを増やしたり減らしたり、操作するときのモード。外枠が青色。

「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

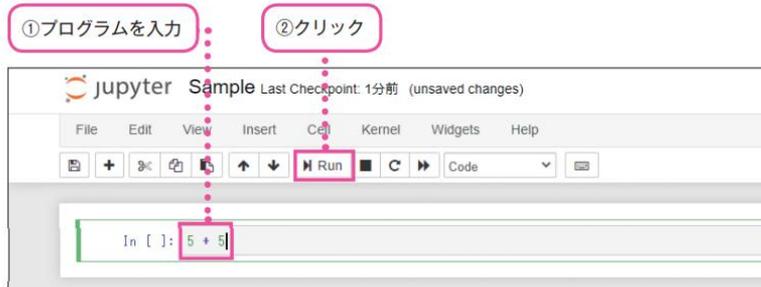
実際に動作の確認をしてみます。

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。



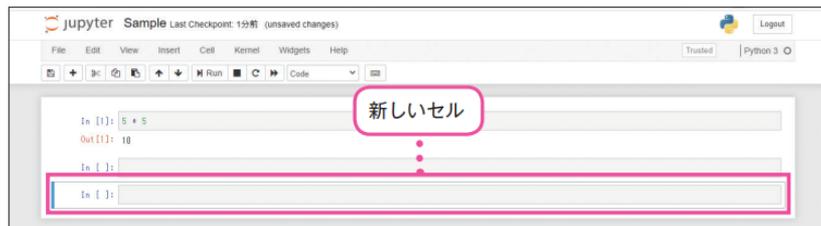
実行後、自動的に新規セルが作成されます。  
この状態で入力欄以外をクリックするか、Escキーを押すと、コマンドモードに切り替わります



# 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

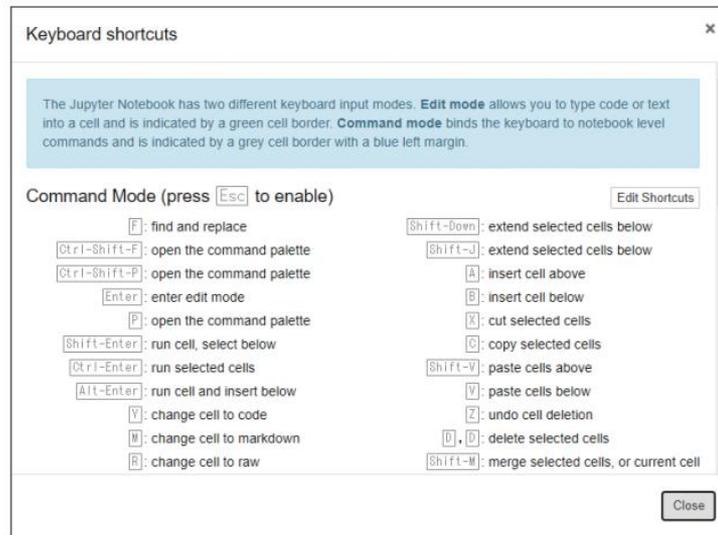


今度は D キーを2回押すと、セルが1つ削除されます。

● 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



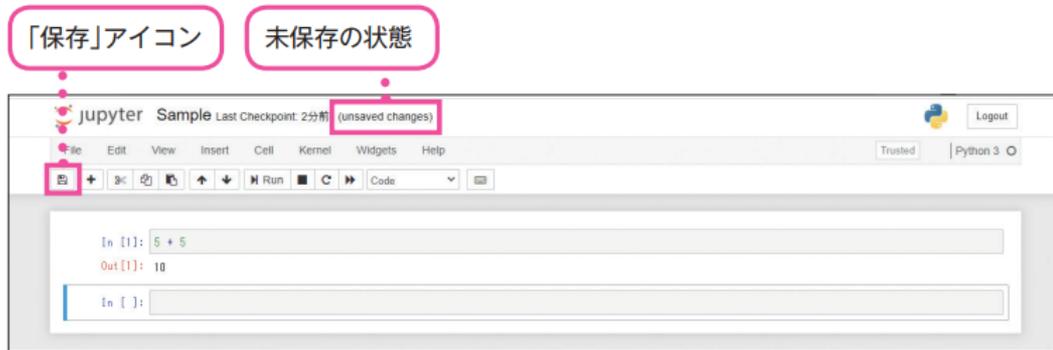
# Jupyter Notebook

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。



# Google Colaboratory

## 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  | Ctrl+M ⇒ B    |
| セルを削除する              | コマンドモード  | Ctrl+M ⇒ D    |
| 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  | Ctrl+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

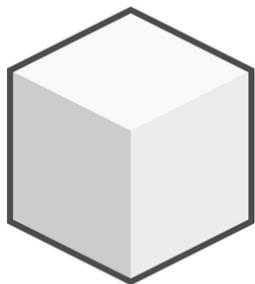
エラーから逃げるべからず

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

$$500 - 100 = 400\text{円}$$

$$500 - 450 = 50\text{円}$$

$$500 - 200 = 300\text{円}$$

## ▼ 変数を使う場合

変数aを500とする。(変数の代入)

$$a - 100 = 400\text{円}$$

$$a - 450 = 50\text{円}$$

$$a - 200 = 300\text{円}$$

# 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

$$500 - 100 = 400\text{円}$$

$$500 - 450 = 50\text{円}$$

$$500 - 200 = 300\text{円}$$

## ▼ 変数を使う場合

変数aを500とする。(変数の代入)

$$a - 100 = 400\text{円}$$

$$a - 450 = 50\text{円}$$

$$a - 200 = 300\text{円}$$



支払い金額が  
1000円の場合は...?

# 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

$$500 - 100 = 400\text{円}$$

$$500 - 450 = 50\text{円}$$

$$500 - 200 = 300\text{円}$$



支払い金額が  
1000円の場合は...?

$$\underline{1000} - 100 = 900\text{円}$$

$$\underline{1000} - 450 = 550\text{円}$$

$$\underline{1000} - 200 = 800\text{円}$$

## ▼ 変数を使う場合

変数aを500とする。(変数の代入)

$$a - 100 = 400\text{円}$$

$$a - 450 = 50\text{円}$$

$$a - 200 = 300\text{円}$$

変数aを1000とする。

$$a - 100 = 900\text{円}$$

$$a - 450 = 550\text{円}$$

$$a - 200 = 800\text{円}$$

# 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

$$500 - 100 = 400\text{円}$$

$$500 - 450 = 50\text{円}$$

$$500 - 200 = 300\text{円}$$

変数を使ったほうが変更箇所が少なく済む!

1000円の場合は...?

$$\underline{1000} - 100 = 900\text{円}$$

$$\underline{1000} - 450 = 550\text{円}$$

$$\underline{1000} - 200 = 800\text{円}$$

## ▼ 変数を使う場合

変数aを500とする。(変数の代入)

$$a - 100 = 400\text{円}$$

$$a - 450 = 50\text{円}$$

$$a - 200 = 300\text{円}$$

変数aを1000とする。

$$a - 100 = 900\text{円}$$

$$a - 450 = 550\text{円}$$

$$a - 200 = 800\text{円}$$

# 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)

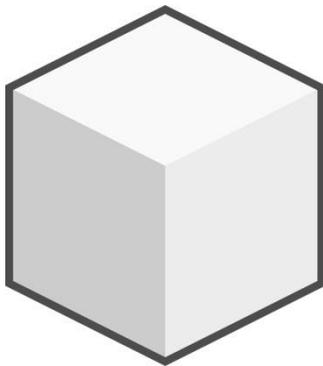


50

※ 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 | micelin  | 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



とある人物Aさん

付属情報) 高さ → 170.0cm (身長)  
血液型 → O型

できること) 食事を摂る → ○

▼「自動車」型



Aさんが所持する自動車B

高さ → 205.0cm (車高)  
血液型 → ×

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)

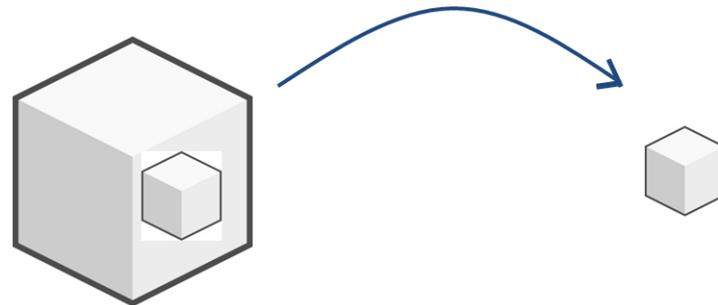
変数trainはDataFrame型のオブジェクト。  
DataFrame型のオブジェクトは、  
**shape**という名前で行数と列数を保持している。

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.>

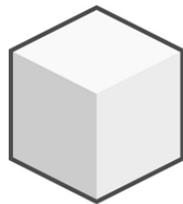
train.shape

↑  
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



・ 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



とある人物Aさん

付属情報)

A.height → 170.0 (身長)  
A.blood\_type → O型

できること)

A.eat() → 食べる

▼「自動車」型



Aさんが所持する自動車B

B.height → 205.0 (車高)  
B.blood\_type → ✕

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一覧

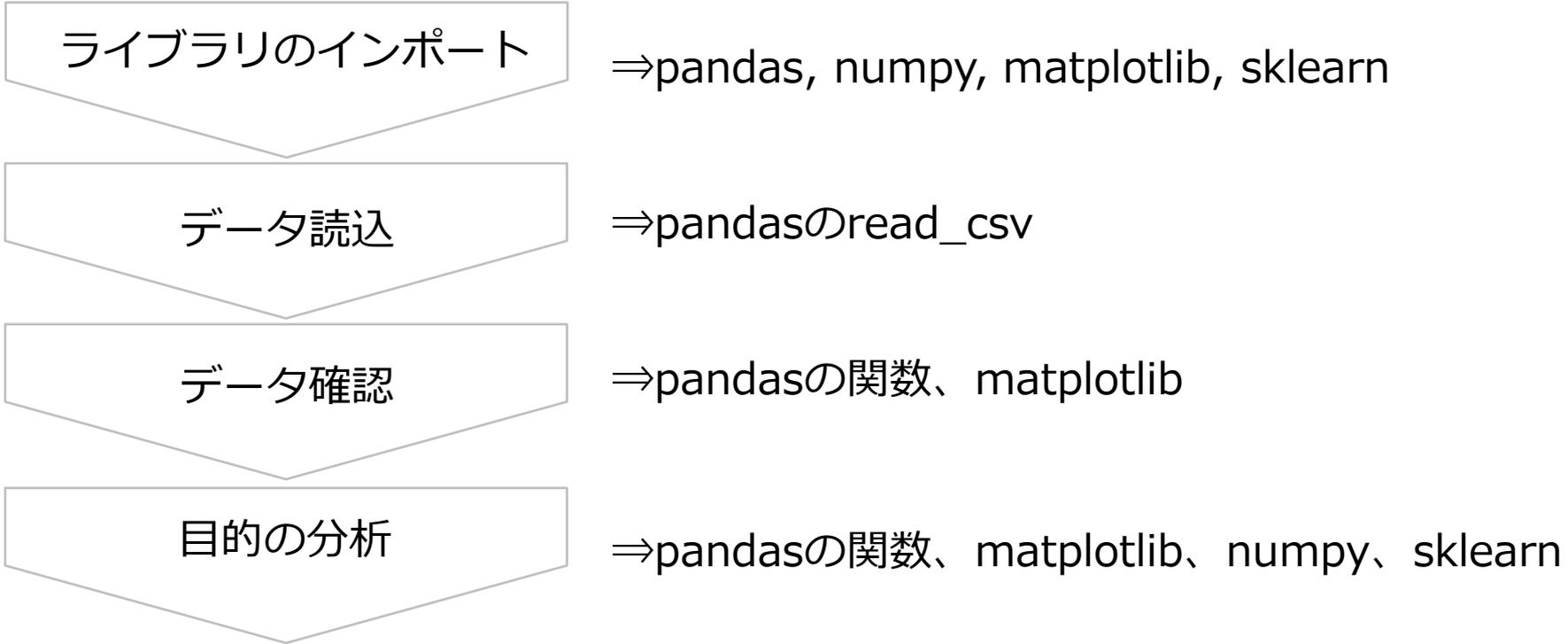
Pythonでデータ分析を行う上で欠かせないライブラリの一覧

- **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



# 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

ライブラリのインポート

⇒pandas, numpy, matplotlib, sklearn

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

データ確認

⇒pandasの関数、matplotlib

目的の分析

⇒pandasの関数、matplotlib、numpy、sklearn

# 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(下記はどちらも同じです)
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 | micelin  | 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")

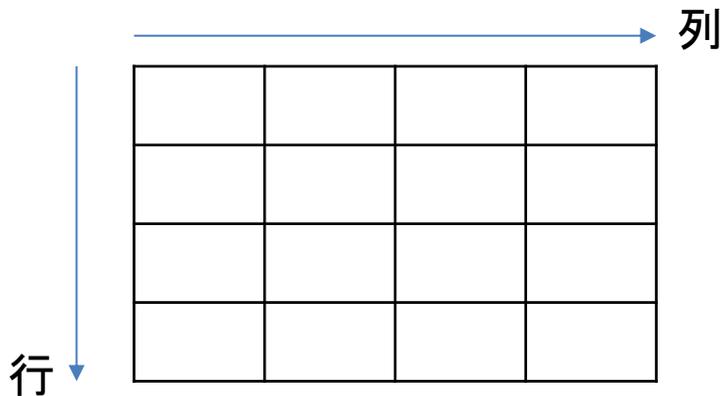
行数列数の確認
data.shape
```

Out[3]:

(1990, 6)

行数

列数



# 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



エラー

# 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**
  - descri**b**eをdescri**V**eと書いてしまう
- **大文字・小文字のミス**
  - **F**alseを**f**alse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)

エラーの原因となっているプログラムの場所

NameError: name 'z' is not defined

エラーの理由

エラー名

# 代表的なエラー

- ModuleNotFoundError
- Name Error
- Syntax Error
- FileNotFoundError
- Attribute Error

# ModuleNotFoundError

```
import pandas
```

---

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

```
ModuleNotFoundError: No module named 'pandas'
```

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

# FileNotFoundException

```
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

- ・ 〇〇がしたい!

→ 「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
- ○○がしたい！



検索エンジンを上手に利用しましょう

🔍 pandas データ読み込み

ライブラリ名 + 作業内容

🔍 python syntaxerror

Python + エラー名、エラー理由

🔍 python deeplearning ライブラリ

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.nkmm.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！

7 日月火水木金土  
2025 1 2 3 4 5  
6 7 8 9 10 11 12  
13 14 15 16 17 18 19  
20 21 22 23 24 25 26  
27 28 29 30 31

**8** August  
2025

9 日月火水木金土  
2025 1 2 3 4 5 6  
7 8 9 10 11 12 13  
14 15 16 17 18 19 20  
21 22 23 24 25 26 27  
28 29 30

| 日 Sun | 月 Mon         | 火 Tue | 水 Wed | 木 Thu | 金 Fri | 土 Sat |
|-------|---------------|-------|-------|-------|-------|-------|
| 27    | 28            | 29    | 30    | 31    | 1     | 2     |
| 3     | 4             | 5     | 6     | 7     | 8     | 9     |
| 10    | 8月3日(日)@オンライン |       |       |       | 15    | 16    |
| 17    | 18            | 19    | 20    | 21    | 22    | 23    |
| 24    | 25            | 26    | 27    | 28    | 29    | 30    |
| 31    | 1             | 2     | 3     | 4     | 5     | 6     |

### ◎ 次回までの宿題

- **AI入門 (2h)**

⇒ <https://biz.quest.signate.jp/quests/10004>

- **自動車環境性能の改善 (6h)**

⇒ <https://biz.quest.signate.jp/quests/10001>

# おしまい

*Let's Enjoy DataScience!!!*

