

Python プログラミング基礎 付録

九州大学 数理・データサイエンス教育研究センター

全体の構成

- 教材 1
 - インTRODクション～プログラムってなんだ？
 - プログラミング言語 Python ～無料で遊べるのに凄い
 - 【準備】Gogole Colaboratoryへ行こう！
 - 文字を表示してみよう ～ 記念すべき最初のプログラム
- 教材 2
 - 計算させてみよう ～ たす, ひく, かける, わる
 - 変数 ～ 電卓を越える！
 - If文 ～ ますますプログラムっぽく
 - 便利なコメントアウト ～ ちょっと休憩
 - 数当てゲームを作ってみよう ～ ゲームの基本中の基本
- 教材 3
 - For文 ～ 人間には面倒な繰り返しを簡単に
 - 【寄り道】コンピュータの計算精度 ～ 意外と正確じゃない
 - 再びゲームを作ってみよう ～ 工夫次第でどんどん楽しく
- 教材 4
 - List ～ 複数の変数を固めたもの
 - 関数とライブラリ ～ ひみつ道具で遊ぼう
- 付録
 - if文のちょっと進んだ使い方
 - AIを用いたプログラミング
 - Google Driveとの連携

【付録】 if文のちょっと進んだ使い方

and と or (1/3)

- if文は「ある条件を満たすときに〇〇をする」際に使う
- 例：「scoreが60以上なら，合格と表示」

```
▶ score = 80
  if score >= 60:
    print("合格")
  else:
    print("不合格")
```

条件

⇒ 合格

```
▶ score = 40
  if score >= 60:
    print("合格")
  else:
    print("不合格")
```

条件

⇒ 不合格

and と or (2/3)

- では、「**2つの条件を同時に満たす**とき〇〇をする」は？
 - “**and**”を使います！
- 例えば「**mathが60以上**で**historyも60以上**だと合格」の場合

```
▶ math = 70
  history = 80
  if math >= 60 and history >= 60:
    print("合格")
  else:
    print("不合格")
```

両方とも
60以上

2条件

合格

```
▶ math = 70
  history = 50
  if math >= 60 and history >= 60:
    print("合格")
  else:
    print("不合格")
```

historyは
60以上ではない

2条件

不合格

and と or (3/3)

- では、「**2つの条件のどちらかを満たす**とき〇〇をする」は？
 - “or”を使います！
- 例えば「**mathが60以上もしくはhistoryが60以上だと合格**」の場合

```
▶ math = 70
  history = 50
  if math >= 60 or history >= 60:
    print("合格")
  else:
    print("不合格")
```

historyは60以上ではないが、
mathは60以上

2条件

合格

```
▶ math = 40
  history = 50
  if math >= 60 or history >= 60:
    print("合格")
  else:
    print("不合格")
```

どちらも60点以下

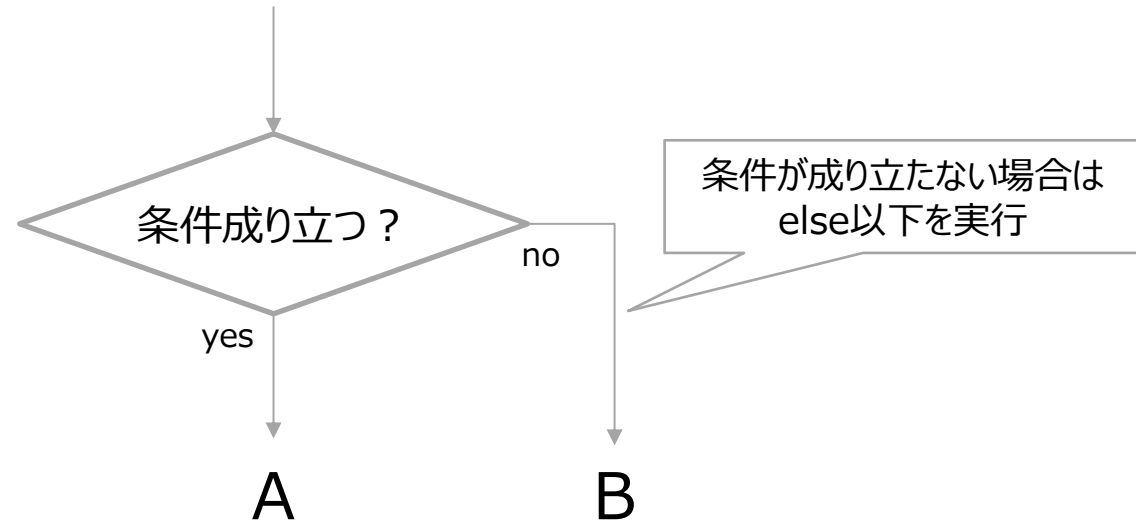
2条件

不合格

- なお、「**どちらも60以上**」でも合格となります

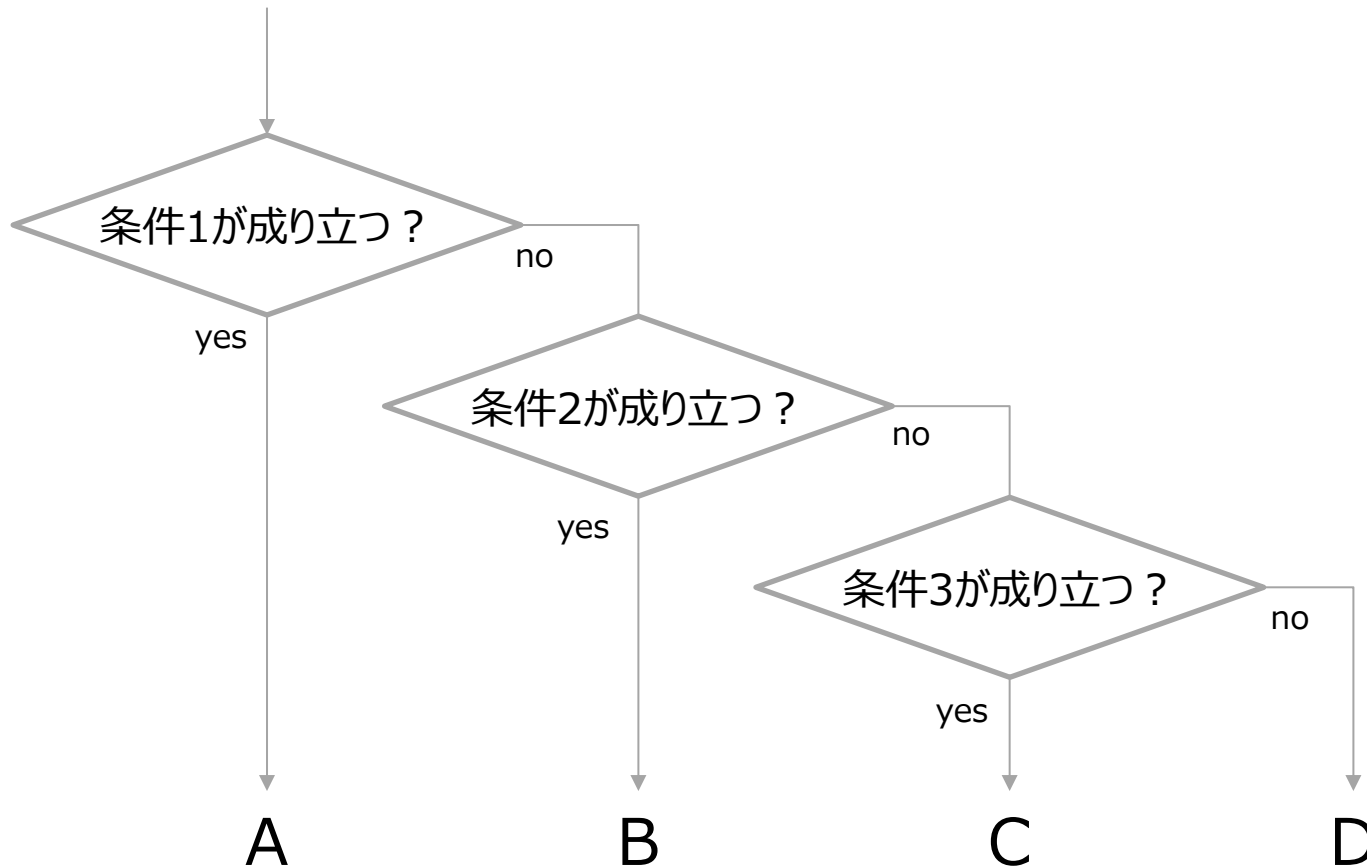
if, else, そしてelif (1/6)

- 「if 条件: A else: B」は、条件を満たすときはAを実行、満たさなかったらBを実行、でした
- これを「フローチャート」で表現します



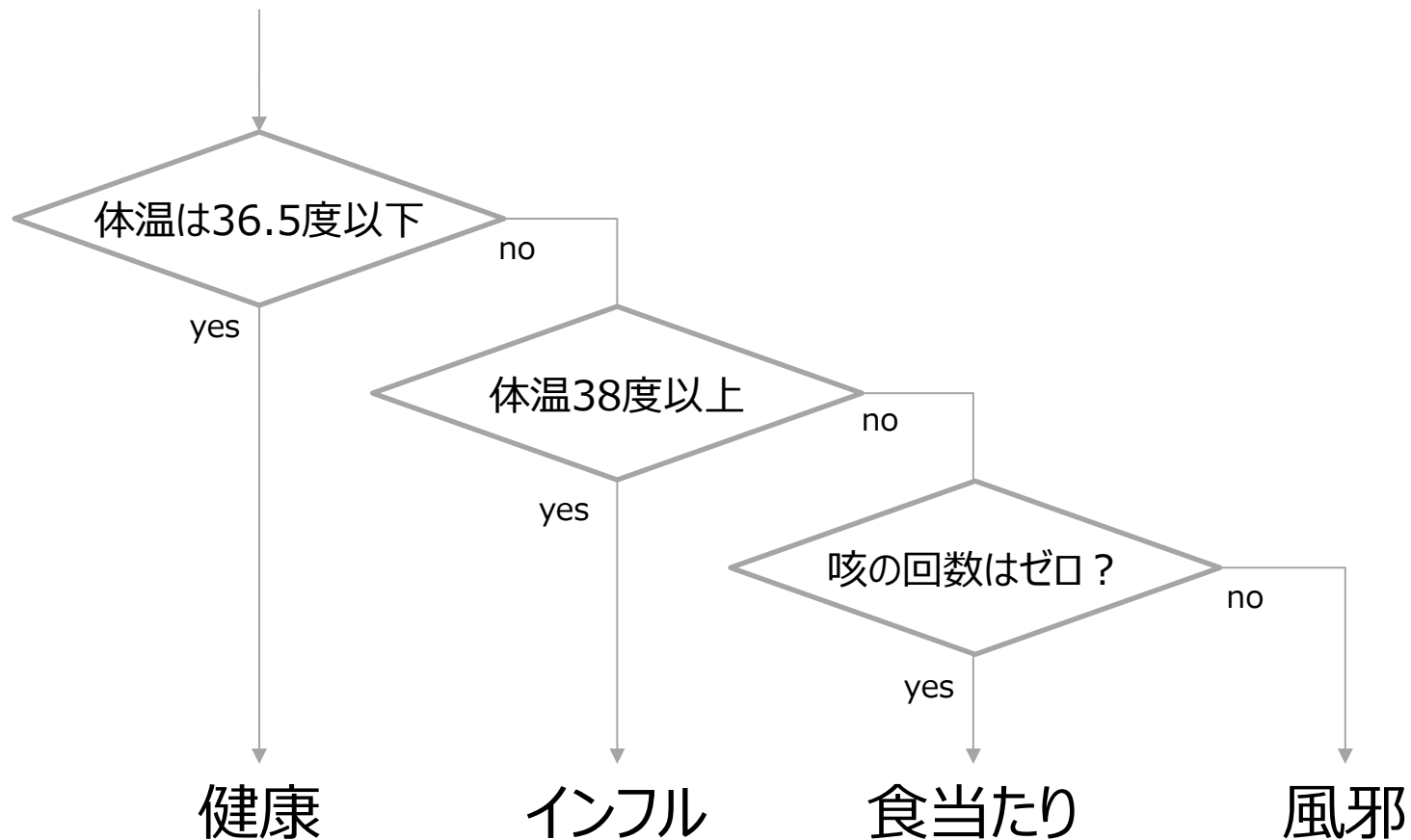
if, else, そしてelif (2/6)

- しかしこんな場合もあるかも…



if, else, そしてelif (3/6)

- 例えば...



if, else, そしてelif (4/6)

- **elif** (else ifの意味)を使えば、前の条件を満たさなかった場合に、別の条件についての判断ができる

```

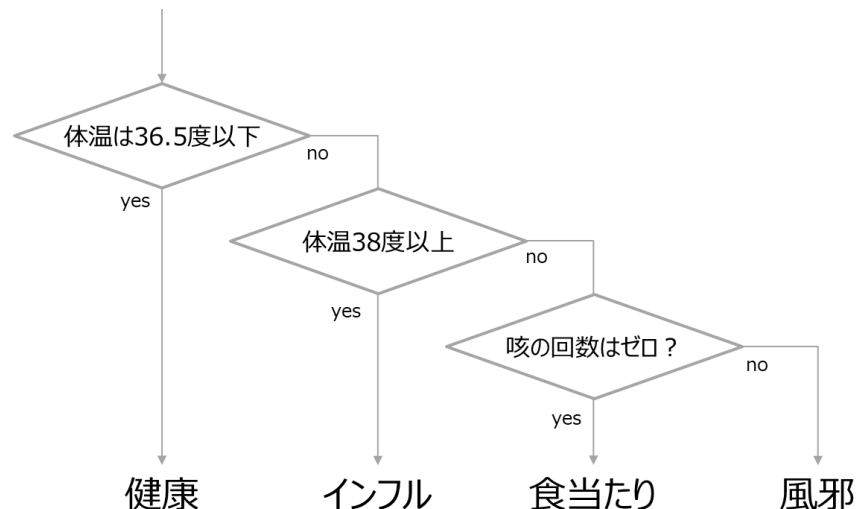
▶ heat = 37.2 #体温
  cough = 0   #咳の回数
  if heat <= 36.5:
    print("健康")
  elif heat >= 38:
    print("インフル")
  elif cough == 0:
    print("食当たり")
  else:
    print("風邪")

```

← elif

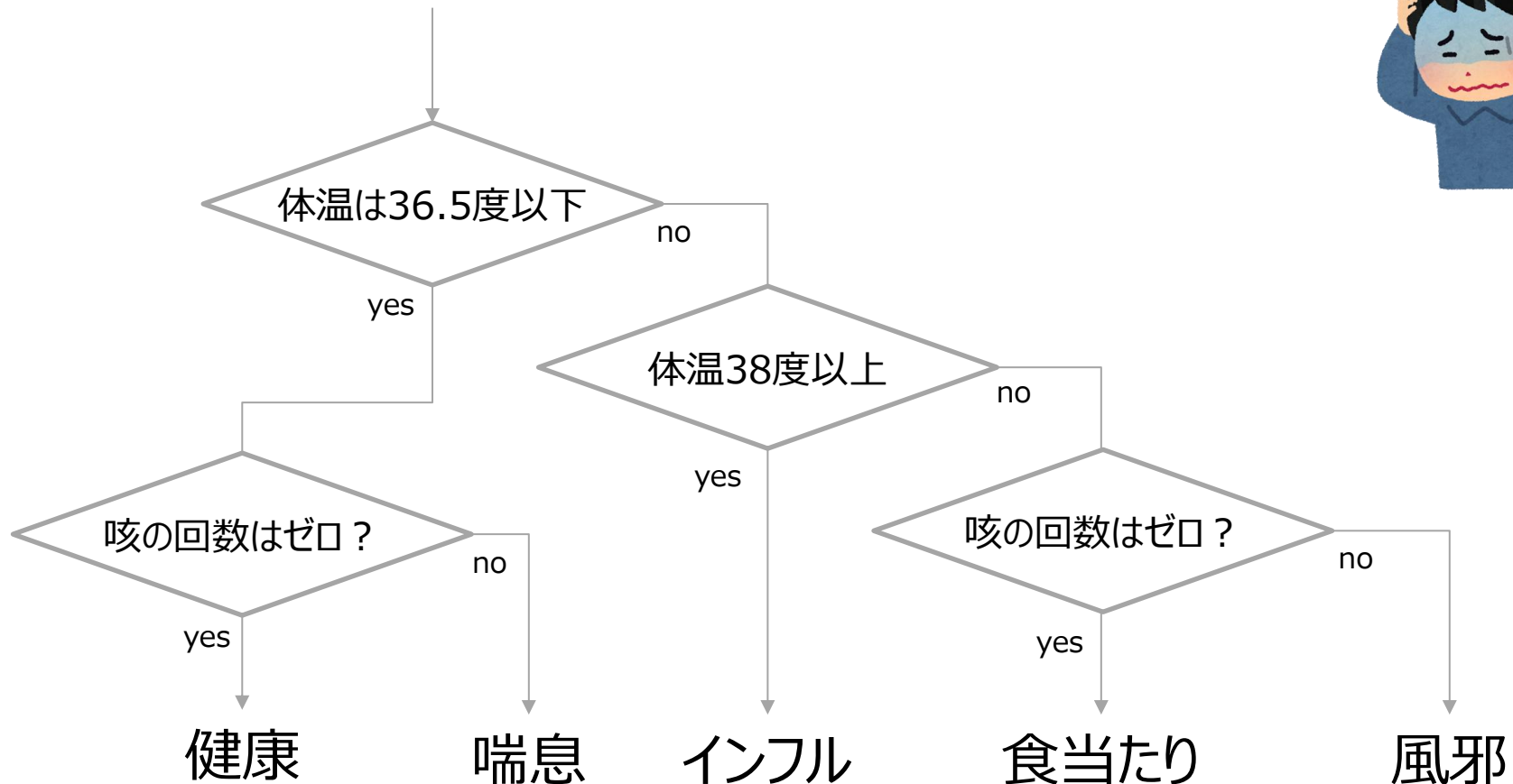
← elif

⇒ 食当たり



if, else, そしてelif (5/6)

- もっと複雑なケースもありそう…さてどうする？



if, else, そしてelif (6/6)

- 複雑にはなりますが，落ち着いて考えればできます！
 - ifの中にifを入れてもよかったことを思い出して…



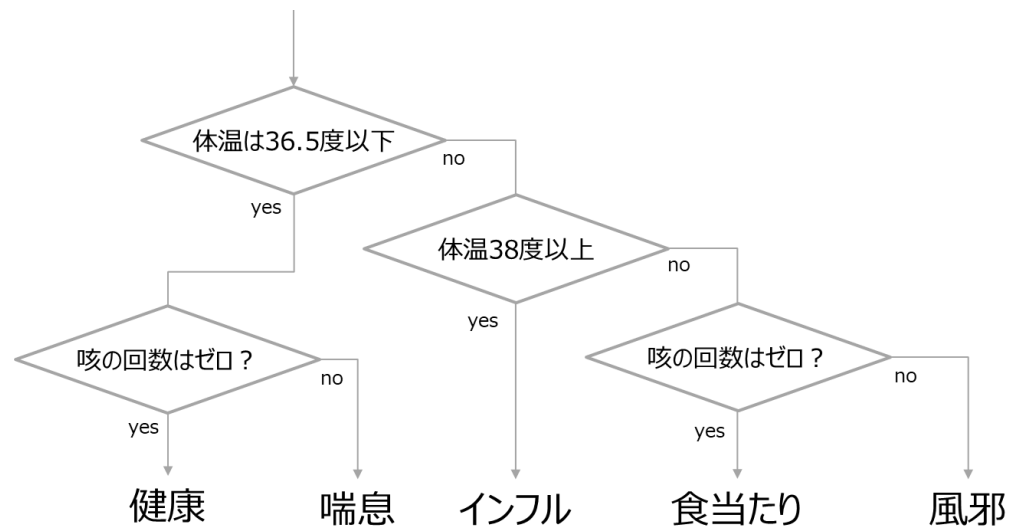
性格判断もこんな感じ？

```

▶ heat = 36.2 #体温
  cough = 2   #咳の回数
  if heat <= 36.5:
    if cough == 0:
      print("健康")
    else:
      print("喘息")
  elif heat >= 38:
    print("インフル")
  elif cough == 0:
    print("食当たり")
  else:
    print("風邪")

```

追加



⇒ 喘息

※この例はもちろん医学的には全く根拠ありません。あくまでプログラムの説明のための例です

- andやorを使うと，よりシンプルに書けるような場合もあります

練習：前に見たこれは..!?

所持金が700円以上ならかつ丼, 500~700円なら親子丼,
400~500円なら素うどん, 400円未満ならごはん抜き!



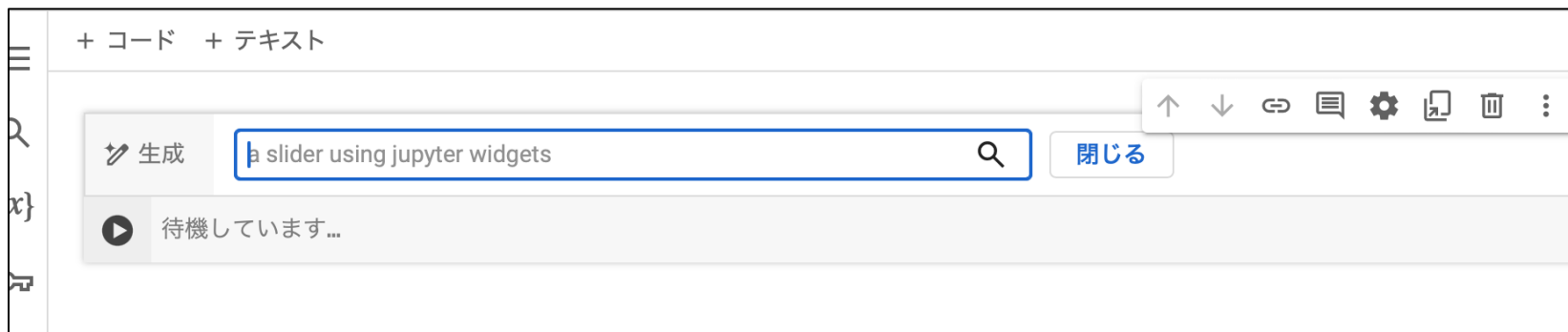
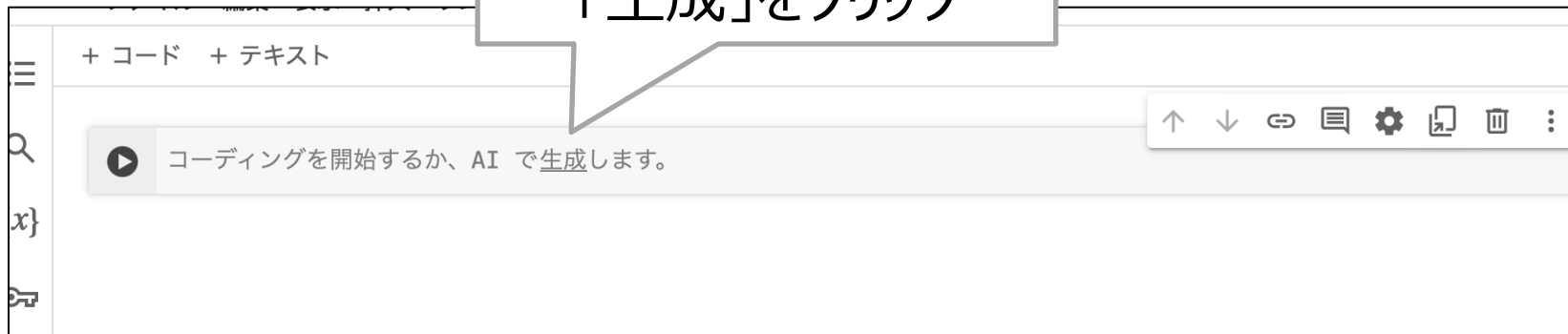
```
▶ money = 580
  if money >= 700:
    print("かつ丼")
  elif money >= 500:
    print("親子丼")
  elif money >= 400:
    print("素うどん")
  else:
    print("ごはん抜き!")
```

⇒ 親子丼

【付録】 AIを用いたプログラミング

AIを使ったプログラミング

「生成」をクリック



AIを使ったプログラミング

- 簡単なプログラムなら自動で作ってくれる

作ってほしい内容を言葉で表現



The screenshot shows a web interface for AI code generation. At the top, there is a search bar with the prompt "サイコロをプログラミングしてください" (Please program a dice). Below the search bar, there is a button labeled "閉じる" (Close). The main area displays the generated Python code, which includes a function to roll a dice and a call to that function. The code is as follows:

```
# prompt: サイコロをプログラミングしてください

import random

def roll_dice():
    """サイコロを振って結果を返す関数"""
    result = random.randint(1, 6)
    return result

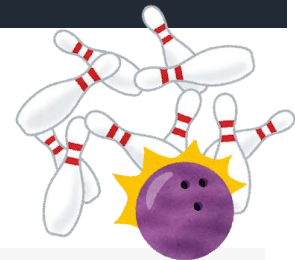
# サイコロを振る
dice_result = roll_dice()
print("サイコロの結果:", dice_result)
```

動作確認は必須

- AIが作ったプログラムを**疑おう**！
 - 間違っているかも（さらに，エラーも出るかも）
 - 自分の指示が曖昧で，想定と違う内容のプログラムが生成されているかも
- 初心者ほど生成結果を入念にチェックしたほうがよい
 - 面倒だが，結果的にはこれが一番実力が付く！
- 「ぱっと見でプログラムの正当性がわかる」や「動作チェックの方法をよく知ってる」レベルの熟練者にとっては非常に便利！
 - そこまで到達すれば，ジャンジャン使おう！



動作確認の必要性の例： 「ボーリングのスコアを自動計算」と入れただけで…



```
# prompt: ボーリングのスコアを自動集計する.
```

```
def calculate_bowling_score(scores):
    """
    ボーリングのスコアを計算する関数.

    Args:
        scores: 各フレームのスコアを格納したリスト.
               例: [[10, 0], [7, 2], [9, 0], ...]

    Returns:
        ... 合計スコア.
    """
    total_score = 0
    frame_index = 0

    for frame in range(10): # 10フレームまでループ
        if frame_index >= len(scores):
            break

        score = scores[frame_index]

        if score[0] == 10: # ストライク
            total_score += 10
            if frame_index + 1 < len(scores):
                total_score += scores[frame_index+1][0]
                if scores[frame_index + 1][0] == 10: # 次のフレームもストライク
                    if frame_index + 2 < len(scores):
                        total_score += scores[frame_index + 2][0]
                    else:
                        total_score += 0 # 次のフレームがない場合は0点追加
            else:
                total_score += scores[frame_index+1][1]
            frame_index += 1
        elif score[0] + score[1] == 10: # スペア
            total_score += 10
            if frame_index + 1 < len(scores):
                total_score += scores[frame_index+1][0]
            frame_index += 1
        else:
            total_score += score[0] + score[1]
            frame_index += 1

    return total_score

# 使用例
scores = [[10, 0], [7, 2], [9, 0], [10, 0], [0, 8],
          [6, 3], [7, 3], [10, 0], [9, 1], [8, 1]] # 例のスコア
total_score = calculate_bowling_score(scores)
print(f"合計スコア: {total_score}")

scores = [[10, 0], [10, 0], [10, 0], [10, 0], [10, 0],
          [10, 0], [10, 0], [10, 0], [10, 0], [10, 10, 10]] # パーフェクト
total_score = calculate_bowling_score(scores)
print(f"合計スコア: {total_score}")
```

こんな長い
プログラムが
一瞬で生成される

しかし…

```
# 使用例
scores = [[10, 0], [7, 2], [9, 0], [10, 0], [0, 8],
          [6, 3], [7, 3], [10, 0], [9, 1], [8, 1]] # 例のスコア
total_score = calculate_bowling_score(scores)
print(f"合計スコア: {total_score}")

scores = [[10, 0], [10, 0], [10, 0], [10, 0], [10, 0],
          [10, 0], [10, 0], [10, 0], [10, 0], [10, 10, 10]] # パーフェクト
total_score = calculate_bowling_score(scores)
print(f"合計スコア: {total_score}")
```

合計スコア: 139
合計スコア: 270

これは正しいが…

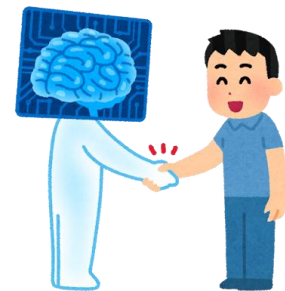
これはおかしい！ボーリングのパーフェクトは300！

本当は、9フレームや10フレームに
ストライクが出た時のための例外処理が
必要なのに、それがプログラムされていない！

叱られないエラーだから余計怖い！

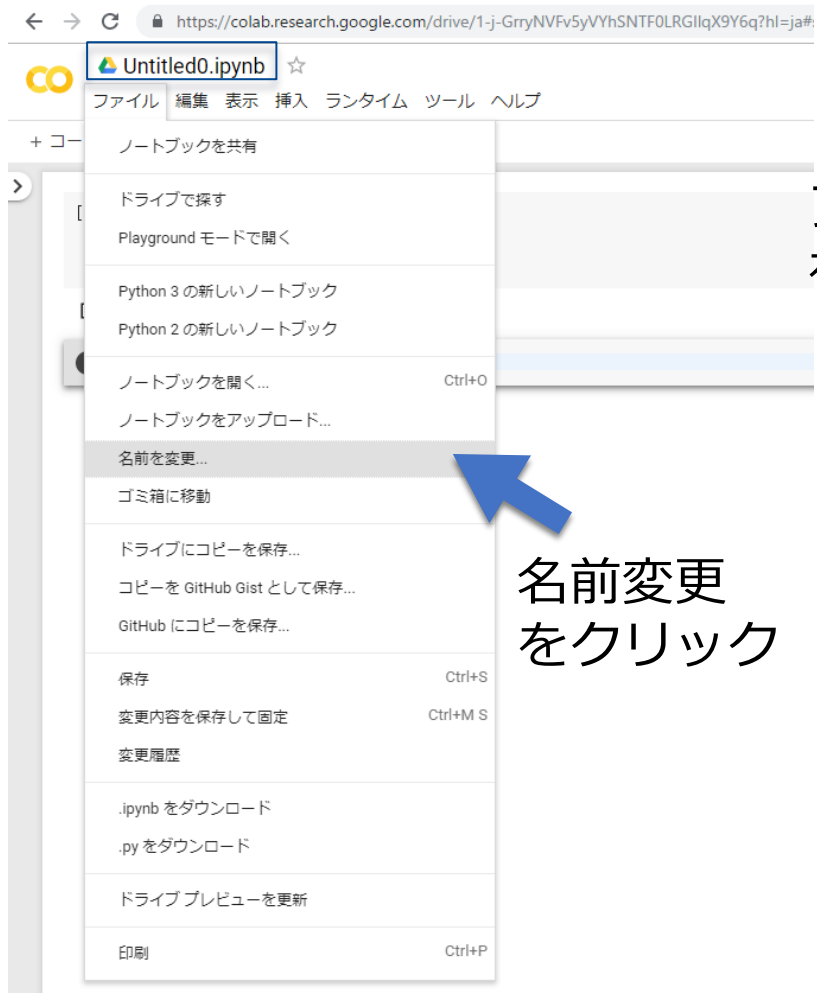
講義での利用

- 使う分には全然構いません
- 練習プログラムをAIに書かせただけではコピーと同じ
 - それを信じて「動いた！」と喜ぶだけだと、自分の勉強にはならない
- 「学習・理解のサポーターとして適切に使う」ぐらいがよい
 - 例えば、最初は自分の力 (+TAへの質問) で頑張る、その後AIの生成と比較してみる、なども



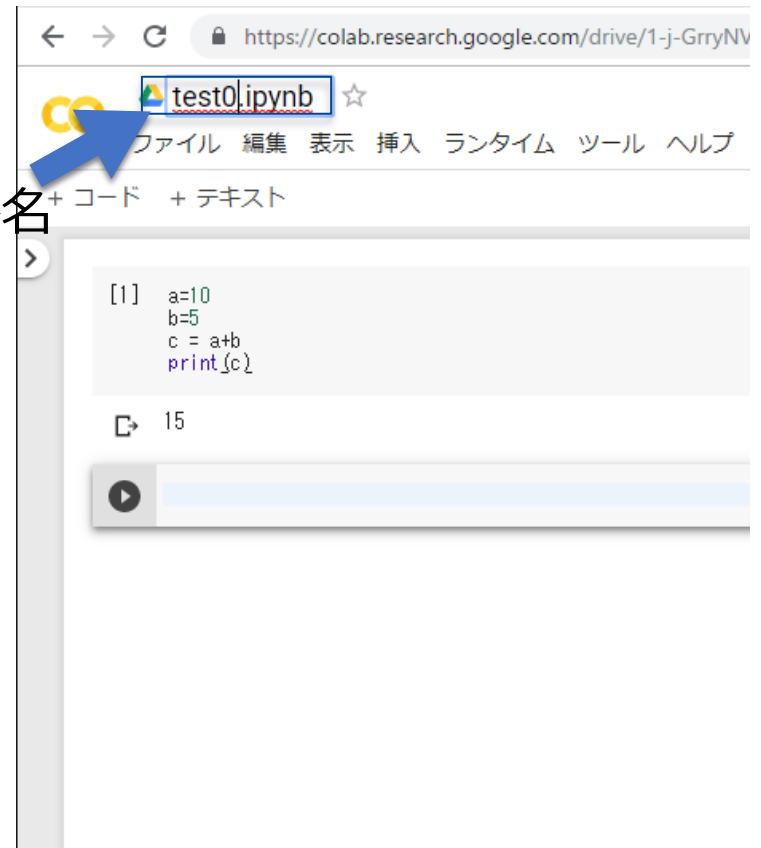
【付録】 Google Driveとの連携

ファイル名を変更

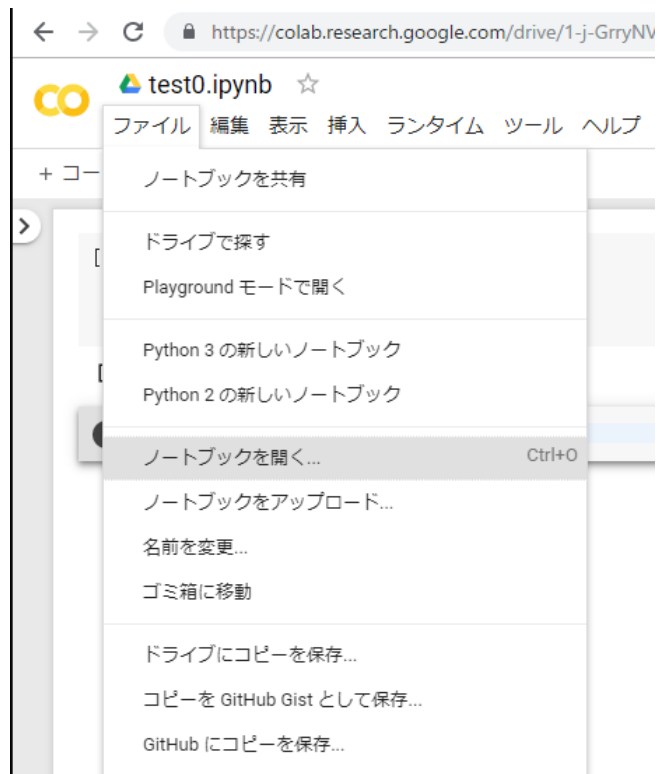


ファイル名
を編集

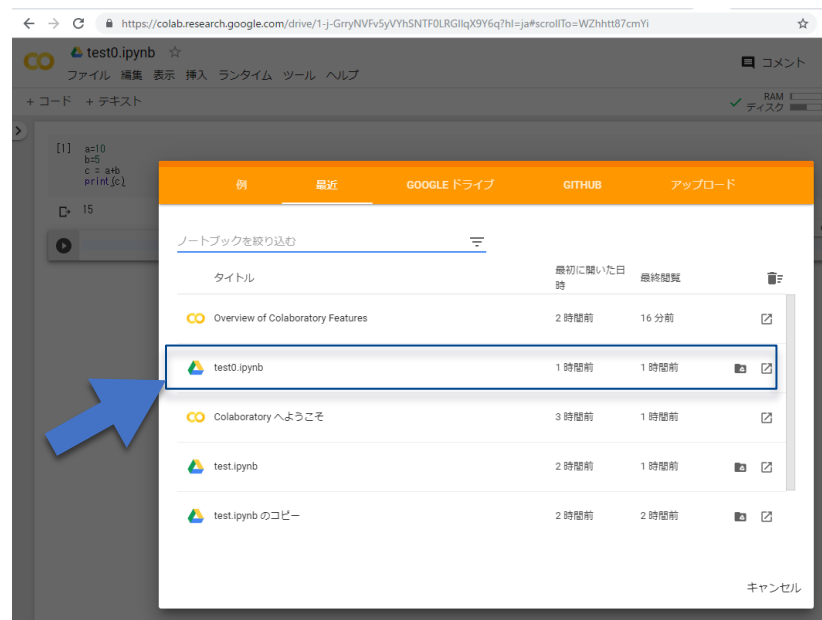
名前変更
をクリック



ファイルを開く



さっきの
ファイルが
選択できる



ファイルの保存先 : Google Drive

Drive

Search in Drive

New

My Drive

My Drive trash is changing. in your trash for 30 days. [Learn more](#)

Quick Access

データサイエンス実践特別講座 申し込み受付

データサイエンス実践特別講...
末廣大貴 edited in the past year

Folders

Colab Notebooks

自動的に同じアカウントの
Google Drive の下に
フォルダが作成されている

Drive

Search in Drive

New

My Drive > Colab Notebooks

My Drive trash is changing. Sta in your trash for 30 days. [Learn mor](#)

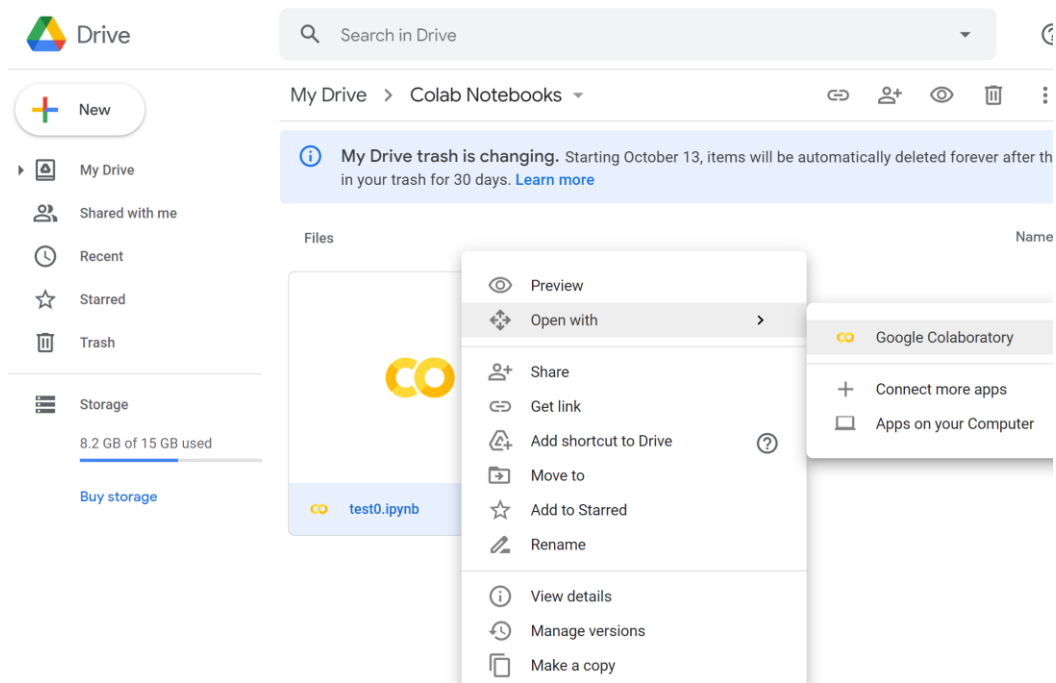
Files

CO

test0.ipynb

先ほどのファイルが保存されている

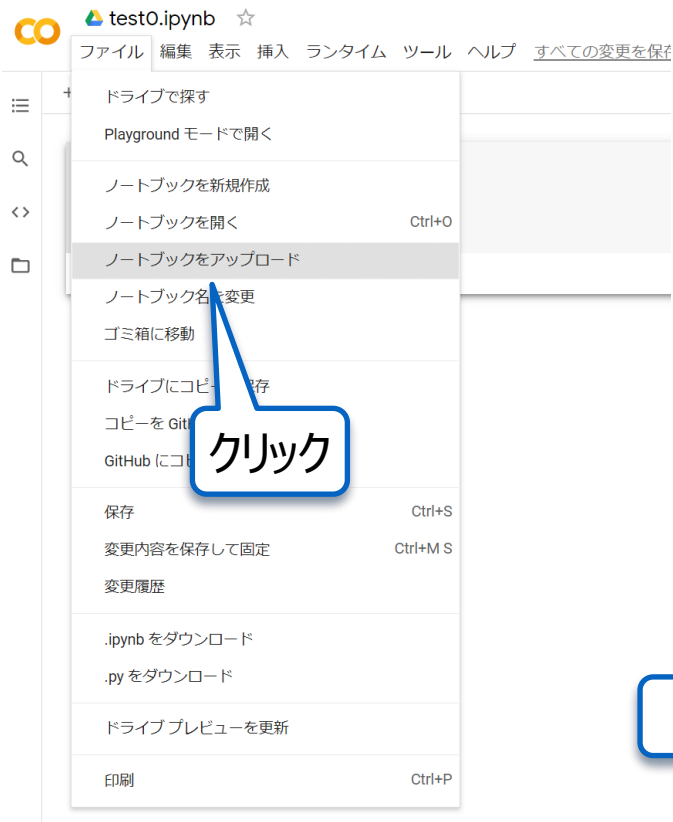
ファイルの保存先 : Google Drive



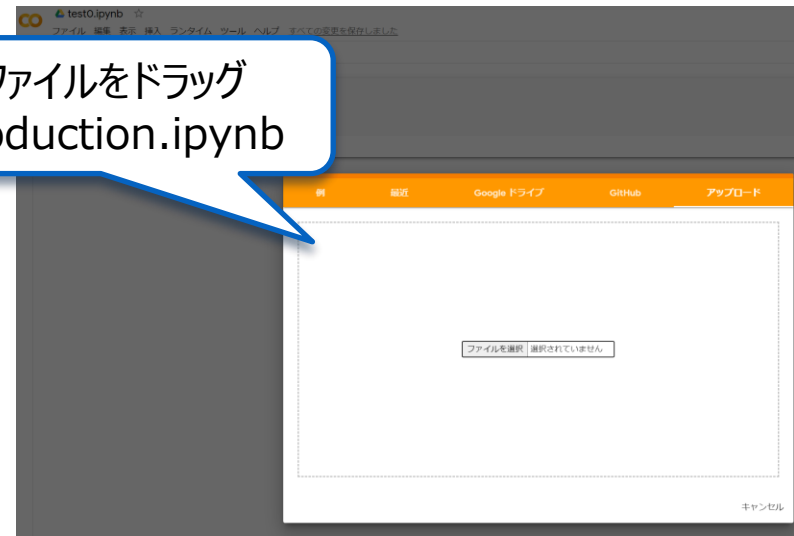
Google Driveから
ファイルを開ける

ファイルの上で右クリックして、
Open with -> Google Colaboratory をクリック

Fileのアップロード



PCからファイルをドラッグ
例 : introduction.ipynb



Introduction.ipynbが開く



DriveへのFileアップロード

My Drive > Colab Notebooks ▾

My Drive trash is changing. Starting October 13, items will be automatically deleted in your trash for 30 days. [Learn more](#)

Files

- introduction.ipynb
- test0.ipynb

Drop files to instantly upload them to:
Colab Notebooks

Get Drive for desktop
Download Learn more

Drive

Search in Drive

My Drive > Colab Notebooks ▾

- basis_of_python_1.ipynb
- basis_of_Python_2.ipynb
- clustering.ipynb
- correlation.ipynb
- imageprocessing1.ipynb
- imageprocessing2.ipynb
- introduction.ipynb
- MNIST.ipynb
- PCA.ipynb
- statistics.ipynb
- test0.ipynb

Get Drive for desktop
Download Learn more

複数ファイルまとめてDrop

新しいフォルダの作成

basis_of_python_1.ipynb ☆
ファイル 編集 表示 挿入 ランタイム ツール ヘルプ すべての変更を保存しました

ファイル

- bin
- boot
- content
 - sample_data
 - data**
- datalab
- dev
- etc
- home
- lib
- lib32
- lib64
- media
- mnt
- opt
- proc
- root

+ コード + テキスト

```
[1] # list
a = [0, 1, 2]
print(a)

b = [2, 0.5, 4, 2, 1]

print(c)

e = [{"A", 52, 1.7}, {"B", 75, 1.6}]
print(e[1][1])
```

アップロード
フォルダの削除
ファイル名の変更
新しいファイル
新しいフォルダ
パスをコピー
更新

Contentの下に新しい
フォルダ'data'を作成

フォルダができた

basis_of_python_1.ipynb ☆
ファイル 編集 表示 挿入 ランタイム ツール ヘルプ

ファイル

- bin
- boot
- content
 - data**
 - sample_data
- datalab
- dev
- etc
- home
- lib
- lib32
- lib64
- media
- mnt

+ コード + テキスト

```
# list
a = [0,
print(

b = [2,
print(

b = [2,
print(
print(

list=[
list.ap
list.ap
print(

c = ["+
print(c
```

データのアップロード

basis_of_python_1.ipynb ☆

ファイル 編集 表示 挿入 ランタイム ツール ヘルプ すべての変更を保持

ファイル

- bin
- boot
- content
- data
 - sample_data
- datalab
- dev
- etc
- home
- lib
- lib32
- lib64
- media
- mnt
- opt

+ コード + テキスト

```
# list
a = [0, 1, 2]
print(a)

b = [2, 0.5, 4, 2, 1]
print(b)
```

アップロード

- フォルダの削除
- ファイル名の変更
- 新しいファイル
- 新しいフォルダ
- パスをコピー
- 更新

File Explorer: デスクトップ > 九大 > 講義 > DS講義 > 概論2+演習1_講義資料2020年度版 > Data

名前	更新日時	種類	サイズ
ais.csv	2019/08/16 8:35	Microsoft Excel CS...	15 KB
boston_house.csv	2019/08/16 8:35	Microsoft Excel CS...	62 KB
data.csv	2019/08/16 8:35	Microsoft Excel CS...	12 KB
1.Introduction	2019/08/16 8:35	Microsoft Excel CS...	1 KB
VisualElements	2019/08/16 8:35	Microsoft Excel CS...	1 KB
校閲	2019/08/16 8:35	Microsoft Excel CS...	1 KB
挑戦的萌芽研究	2019/08/16 8:35	Microsoft Excel CS...	4 KB
LAN DISK	2019/08/16 8:35	Microsoft Excel CS...	1 KB
Creative Cloud File	2019/08/16 8:35	Microsoft Excel CS...	1 KB
OneDrive - Kyushu	2019/08/16 8:35	Microsoft Excel CS...	1 KB
PC	2019/08/16 8:35	Microsoft Excel CS...	1 KB
3D オブジェクト	2019/08/16 8:35	Microsoft Excel CS...	1 KB
ダウンロード	2019/08/16 8:35	Microsoft Excel CS...	2 KB
デスクトップ	2019/08/16 8:35	Microsoft Excel CS...	1 KB
ドキュメント	2019/08/16 8:35	Microsoft Excel CS...	10 KB
temperature_data.csv	2019/08/16 8:35	Microsoft Excel CS...	10 KB
temperature_data_Meas...	2019/08/16 8:35	Microsoft Excel CS...	10 KB

ファイル名(N): "grade_data3.csv" "height_weight.csv" "height_weight_small.csv" "height_weight_small2.csv" "height_weight_small3.csv" "iris.csv" "iris_2metrics1.csv" "iris_2metrics2.csv" "laborfor.csv" "predicti.csv" "some.csv" "temperature_data.csv"

すべてのファイル (*.*)

開く(O) キャンセル

```
print(e[1][1])
print(e)
e[0].append(27)
print(e)
```

Data フォルダを選択してアップロードをクリック

配布資料のDataフォルダのパスを指定して、全てのファイルを選択

これでプログラムからファイルを読めるようになった

※) ただし接続が切れると初期化されるので、毎日アップする必要がある

準備 : Google Driveへフォルダごとアップ

The screenshot shows the Google Drive web interface. On the left is a navigation sidebar with options: 'New', 'My Drive' (selected), 'Shared with me', 'Recent', 'Starred', 'Trash', and 'Storage' (showing 8.2 GB of 15 GB used). The main area displays 'My Drive' with a notification about trash changes. Below that is a 'Quick Access' section with three items: 'basis_of_python_1.ipynb' (edited today), 'introduction.ipynb' (edited today), and 'imageprocessing1.ipynb' (uploaded today). At the bottom is a 'Folders' section with three folders: 'Colab Notebooks', 'Google Buzz', and 'ImgData'.

フォルダごとドラッグすれば、
フォルダの中身も全てアップロード可能

Google Driveとの接続

Colaboratory へようこそ
ファイル 編集 表示 挿入 ランタイム ツール ヘルプ 変更を保存できません

ドライブ

bin
boot
content
code
introduction.ipynb
data
drive
datalab
web
run.sh
dev
etc
home
lib
lib32
lib64
media
mnt
opt

Colaboratory とは
Colaboratory (略称: Colab) は、コ

- 環境構築が不要
- GPU への無料アクセス

Drive をマウント
をクリック

はじめに
ご覧になっているこのドキュメン
ラクティブな環境です。

Colaboratory へようこそ
ファイル 編集 表示 挿入 ランタイム ツール ヘルプ 変更を保存できませんでした

ドライブにコピー

bin
boot
content
code
introduction.ipynb
data
drive
datalab
web
run.sh
dev
etc
home
lib
lib32
lib64
media

Colaboratory とは
Colaboratory (略称: Colab) は、ブラウザから Python

- 環境構築が不要
- GPU への無料アクセス
- 簡単に共有

Colab は、学生からデータサイエンティスト、AI リサ
 覧ください。下のリンクからすぐに使ってみることもあ

```
from google.colab import drive
drive.mount('/content/drive')
```

Google ドライブをマウントするには、このセルを実行してください。

閉じる

はじめに
ご覧になっているこのドキュメントは静的なウェブペ
ニケーション環境です。

セルが現れるので、実行

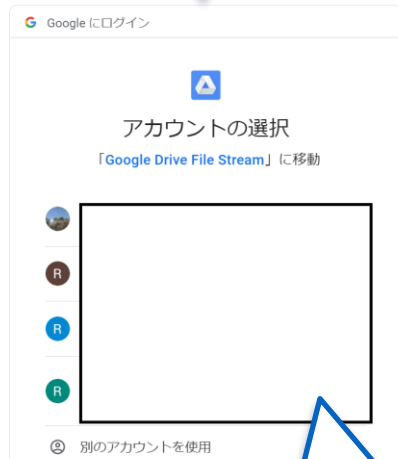
Google Driveとの接続

```
from google.colab import drive
drive.mount('/content/drive')
```

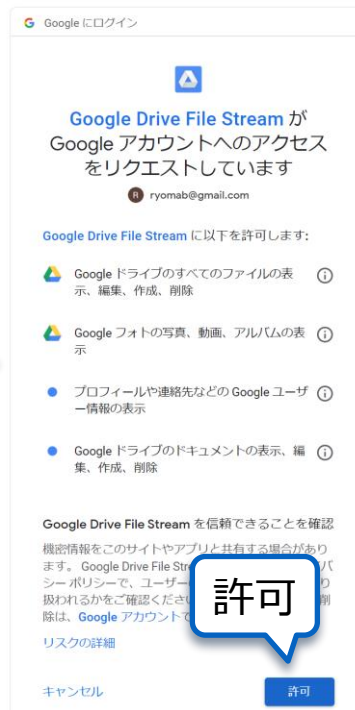
... Go to this URL in a browser: <https://accounts.google.com/o/oauth2/auth?c>

Enter your authorization code:

リンクをクリック



Google Driveの
アカウントを選択



許可



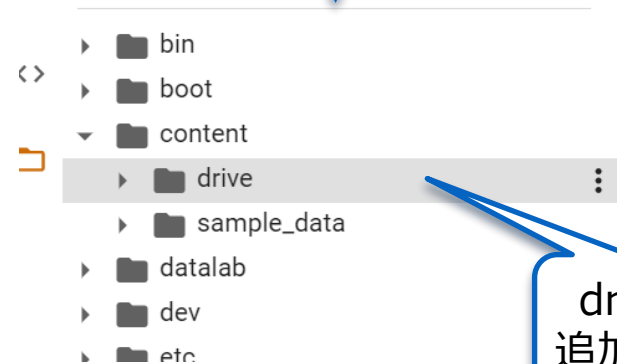
コピー

```
from google.colab import drive
drive.mount('/content/drive')
```

... Go to this URL in a browser: <https://accoun>

Enter your authorization code:

ペースト



driveが
追加された