ae-2. Python プログラムの実行、Python でのデータマネジメント

（ディープラーニング，Python を使用）

（全１５回）

https://www.kkaneko.jp/ai/ae/index.html

金子邦彦

BY NC SA

アドバイス

1. **基本から開始する：**Pythonの**基本的な文法**や**コーディングスタイル**をしっかり学ぶ。
2. **自主性を発揮する：実際のPythonプログラムを読み**、理解を進めましょう。分からないところは**自分で調べたり**、**質問**したり、**試して**みることも重要です。
3. **オンラインサービスの活用**: Trinket や、**Google Colaboratory**などのオンラインサービスも利用して、学習を進めよう。
4. **エラーは学びの機会**: プログラミングで、**エラー**が発生することはよくあります。**エラーメッセージを理解し、問題を解決するスキル**も大切です。エラーを体験するために、自主的な実践が必要です。
5. **学習の継続: AIに興味を持ち続け**、学習を続けましょう。

① 視野の広がりと実力の成長

② 人工知能、IT分野での問題解決能力の向上

③ 将来のキャリアとの関連

アウトライン

1. Python の基本的な文法
2. Python の基本的な機能とコーディングスタイル
3. Google Coraboratory の活用
4. Google アカウント
5. Google Colaboratory の本格的な機能 （Google アカウントが必要）
6. Python の pandas データフレーム
7. Google Colaboratoryを活用してデータを取り扱う実践的な演習

2-1. Python の基本的な文法

Python

- **Python** は多くの人々に利用されている**プログラミング言語**の１つ
- **読みやすさ，書きやすさ，幅広い応用範囲**が特徴

```
from keras.models import Sequentia
model = Sequential()
from keras.layers import Dense, Ac

model.add(Dense(units=64, input_di
model.add(Activation('relu'))
model.add(Dense(units=max(set(y_tr
model.add(Activation('softmax'))
model.compile(loss='sparse_categor
              optimizer='sgd',
              metrics=['accuracy']
model.fit(x_train, y_train, epochs
score=model.evaluate(x_test, y_tes
print(score)
model.predict(x_test)
model.summary()
```

```
och 1/200
'3 [==============================] - 0s
 3200
 och 2/200
 3 [==============================] - 0s
 200
 ch 3/200
  [==============================] - 0s
 0
```

Python 言語が広く使用されている理由

文法のシンプルさ

- **Python** は，**直感的で読みやすい**文法

拡張性

- **豊富なライブラリとモジュール**を持ち、**多岐にわたる分野で利用**が可能

柔軟性

- シンプルなスクリプトも、高度なプログラムも作成可能

主な機能

- **基本的な機能**: 変数、データ型、条件分岐（if）、繰り返し（for、while）、関数
- **オブジェクト指向**: クラス、スーパークラス、サブクラス、継承
- **モジュールとパッケージ**: 再利用可能なコード
- **例外処理**: エラーを効果的に処理
- **ファイル操作**: ファイルの作成、読み出し、書き込み
- **表示**: テキストでの表示、ビジュアルな表示

プログラミングが役に立つ理由

- **プログラミング**は**人間の力を増幅**し、私たちができることを大幅に広げる

- **シミュレーション**、**大量データ処理**、**AI連携**、**ITシステム制作**など、さまざまな活動で、**プログラミング**は役立つ

- **プログラミングはクリエイティブな行為**

- さまざまな**作業を自動化**したいとき、**問題解決**したいときにも役立つ

変数，代入，オブジェクト生成

- **変数**：プログラム内で名前を付けて利用する**オブジェクト**で，**値を保存**し，後から**参照**できる（「変数」は，数学の変数とは違う意味）

- **代入**：プログラムで，「**x = 100**」のように書くことで，**x という名前の変数に、値 100 が保存**される

```
x = 100
```

プログラム

- オブジェクト生成と代入の例

```
x = turtle.Turtle()
```

プログラム

Turtle.Turtle()
turtle パッケージ内の
Turtle クラスのオブジェクト
を生成．x に代入．

オブジェクトとメソッド

- **オブジェクト**：コンピュータでの**操作や処理の対象となるもの**

t.goto(0,100)	**t　オブジェクト** **goto(0,100)　メソッド** 間を「.」で**区切って**いる

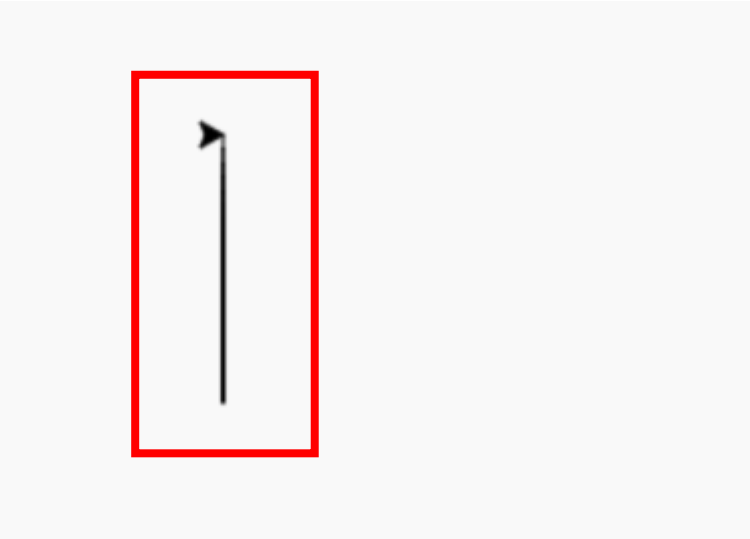
- **メソッド: オブジェクト**に属する機能や操作．オブジェクトがもつ能力に相当する
- **引数：メソッド**が行う操作の詳細に関する情報．**メソッド**呼び出しのときに、引数を指定できる

t.goto(0,100)

引数

オブジェクトとメソッド

オブジェクトが動く



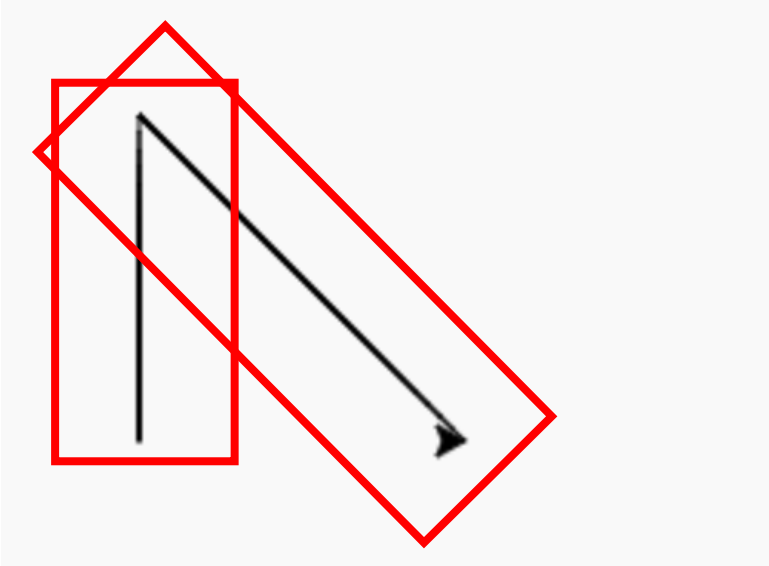
実行画面

```
import turtle
t = turtle.Turtle()
t.goto(0,100)
```

オブジェクトとメソッド
（Python プログラム）

オブジェクトとメソッド

オブジェクトが動く



実行画面

```python
import turtle
t = turtle.Turtle()
t.goto(0,100)
t.goto(100,0)
```

オブジェクトとメソッド
（Python プログラム）

タートルグラフィックス

```
1  import turtle
2  t = turtle.Turtle()
3  t.goto(0,100)
4  t.goto(100,0)
```

オブジェクト　メソッド

- **メソッド**は、オブジェクトが持つ機能を呼び出すためのもの
- 「**goto**」は**指定した座標への移動**

主なメソッド

- **goto**（＜横方向の値＞，<縦方向の値>）　**移動**
- **forward**(＜移動量＞)　**前進**
- **backword**(＜移動量＞)　**後退**
- **right**(＜角度＞)　**右回りに回転**
- **left**(＜角度＞)　**左回りに回転**

Python プログラムの実行（Trinket を使用）

- **trinket** は **Python, HTML などのプログラムを書き実行できる**サイト

https://trinket.io/python/f29bfe71cd

のように、違うプログラムには違う URL が割り当てられる

- 実行が開始しないときは、「**実行ボタン**」で**実行**
- ソースコードを**書き替えて再度実行**することも可能

演習１

ページ１６～１８

【トピックス】

オブジェクト、メソッド、引数

① trinket の次のページを開く

https://trinket.io/python/f29bfe71cd

② 実行結果が，次のように表示されることを確認

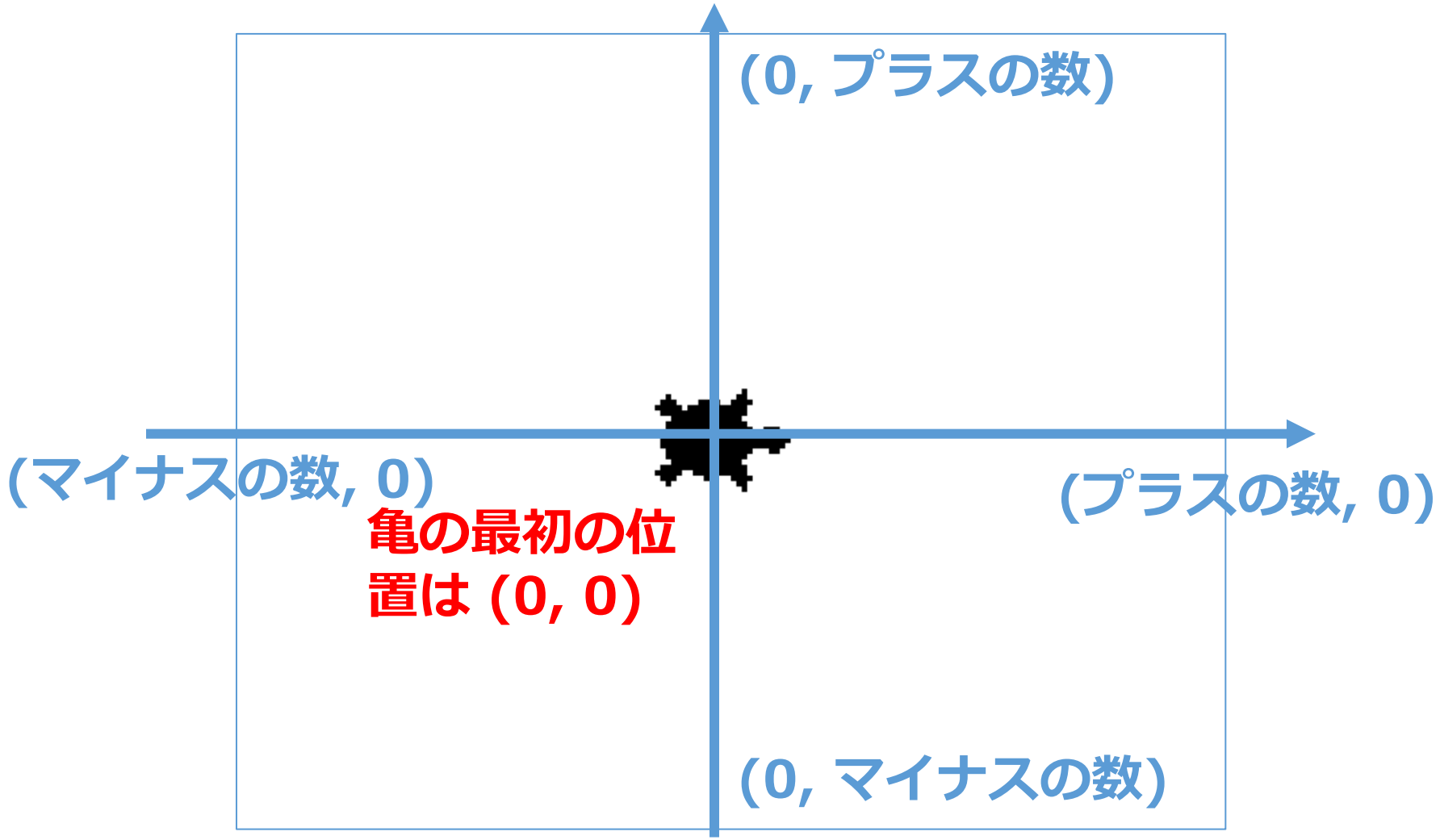
- 実行が開始しないときは、「**実行ボタン**」で**実行**
- ソースコードを**書き替えて再度実行**することも可能

③ 次のように**書き換えて**、**再実行**し、**結果が変わる**ことを確認

```
import turtle
t = turtle.Turtle()
t.goto(0,100)
t.goto(200,0)
```

書き換え

④ 次を参考に、**自分で引数を書き替えたり**、プログラム内に**「t.goto(<引数>)」を増やして**、**思い通りの図形**を目指す。



ここまでのまとめ

- **オブジェクト**：操作や処理の対象となるもの。
- **変数**：**名前を付けて利用するオブジェクト**で，**値を保存**し，**後から参照**できる。
- **代入**：プログラムで変数に値を保存する操作。

「**x = 100**」は**変数x**に**100を代入**する。

- **メソッド**：オブジェクトに属する機能や操作。
- **メソッドアクセス**　：オブジェクト名 + . + メソッド名 + ()
（引数を付けることも）

t.goto(100,0)ではheroはオブジェクト、goto はメソッド。

- **引数**：メソッドが行う操作の詳細情報。

t.goto(100,0) では、引数は「100,0」

2-2. Python の基本的な機能とコーディングスタイル

Python 言語が広く使用されている理由

文法のシンプルさ

- **Python** は，**直感的で読みやすい**文法
- 例えば、**print** で簡単に**出力**できる、**if** や **else** で**条件分岐**、**for** や **while** で**繰り返し（ループ）**

拡張性

- **多岐にわたる分野で利用**が可能
- 例えば、**関数やクラスを定義**するための **def** や **class**、**継承**や**オブジェクトの属性名と値**を操作するための **super** や **vars** などがある。

柔軟性

- シンプルなスクリプトも作成可能、高度なプログラムも作成可能
- **オブジェクト指向**の機能を持ち、**__init__** や **self** のようなキーワードを使用して**クラス**を利用できる。

Python の主なキーワード

- **print**: 出力
- **type**: 型名（クラス名）の取得
- **if**, **else**: 条件分岐
- **for**, **while**: 繰り返し
- **def**: 関数定義
- **return**: 関数の評価値（戻り値）
- **class**: クラス定義
- **__init__**: オブジェクトの初期化（コンストラクタ）
- **self**: クラス内での自オブジェクトへのアクセス
- **vars**: オブジェクトの属性名と値の取得
- **super**: 親クラス（スーパークラス）へのアクセス

Python の特性

動的型付け

- Python では、変数を宣言する際にデータ型の指定が不要
- Java や C++ は、変数の宣言時にデータ型の指定が必要である静的片付け言語

多様な標準ライブラリ

- データ処理、言語処理、Web開発、データベース接続など、多岐にわたる領域でのライブラリを持つ

クロスプラットフォーム

- Python は多くの OS で動作する

Python のコーディングスタイル

インデントによるブロックの区切り

- Python はコードブロックを波括弧（{}）ではなく、インデントで区切る
- 字下げは、通常「**タブ**」**または**「**４つの半角スペース**」の**いずれか**を使用。（字下げをするときは、タブとスペースを**混在させない**ように注意してください）

```
if (x > 20):
  print("big")
else:
  print("small")
```

if での字下げ

空白行

- **プログラムを読みやすく**するために、**空白行**を挿入できる

コメント

- コメントは 、説明を書いたもの
- コメントは、行の先頭に「#」を記述する。
- **コメント**の**記述内容は自由**である。

演習２

資料：２５、２６

【トピックス】

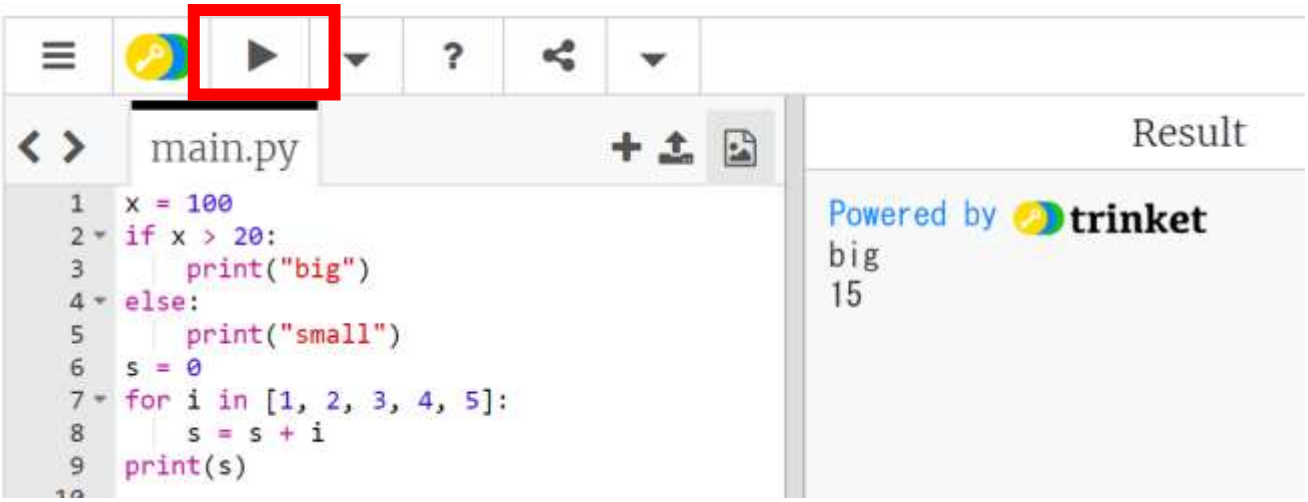
- if, for
- ブロック
- 字下げ

① trinket の次のページを開く

https://trinket.io/python/6c652f1c2f

② 実行結果が，次のように表示されることを確認

実行、STOP ボタン



- 実行が開始しないときは、「**実行ボタン**」で**実行**
- ソースコードを**書き替えて再度実行**することも可能

③ プログラムを**書き換えて実行**し、**結果の変化**を見る（どう書き換えるかは各自で考えてみる）

```
x = 100
if x > 20:
    print("big")
else:
    print("small")
s = 0
for i in [1, 2, 3, 4, 5]:
    s = s + i
print(s)
```

Powered by

```
big
15
```

Python プログラムの基本構造

① Python プログラムの保存と実行

- 通常、Python プログラムを**ファイルに保存する**際の**拡張子は通常「.py」**。
- ファイルに**保存せずに実行することも可能**

② **コメント**はプログラム内で説明を追加するために使用。**コメントは行の先頭に「#」**を書く。

③ 次の基礎を知っておく

変数と代入：「x = 10」

データ型：　整数、浮動小数点数、文字列、リストなど

四則演算：+ - * /

比較演算：== != < > <= >=　　**論理演算**: and or not

条件分岐: if else　**繰り返し**: for while　**関数定義**

④ **条件分岐、繰り返し、関数定義**では、**ブロックの範囲**を指定するために**字下げ**を行う

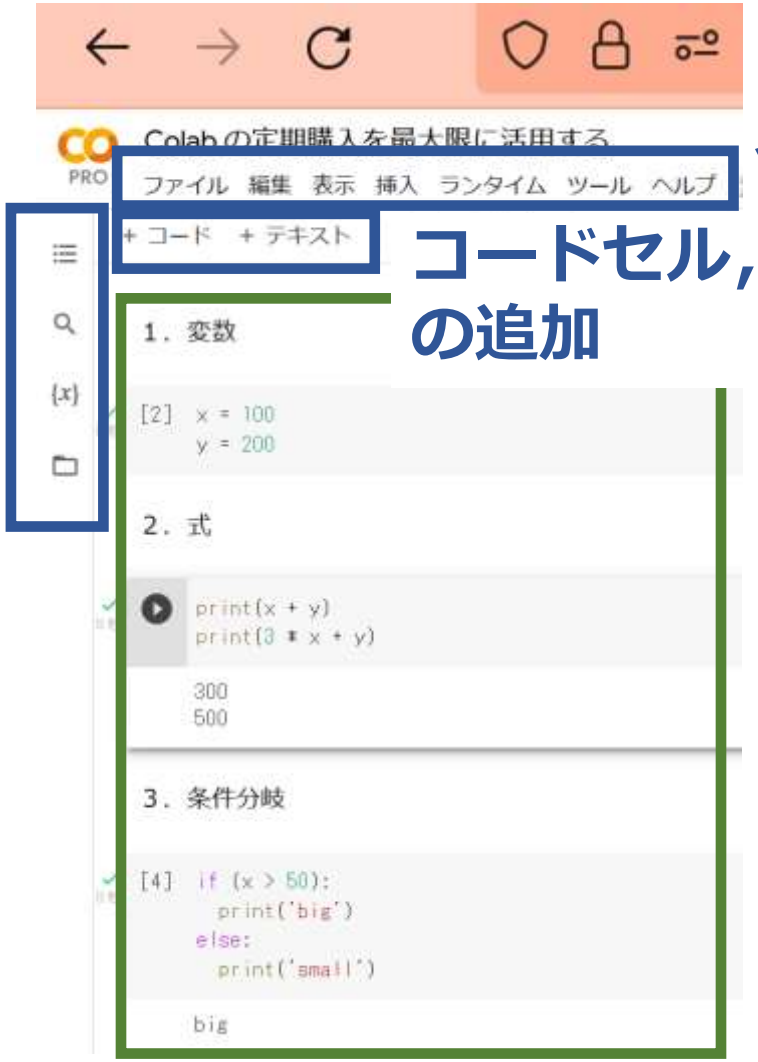
2-3. Google Colaboratory の活用

Google Colaboratory

URL: https://colab.research.google.com/

- **オンラインで動く**
- **Python のノートブック**の機能を持つ
- Python や種々の機能が**インストール済み**
- **本格的な利用**には，**Google アカウントが必要**

Google Colaboratory の全体画面



メニュー

コードセル，テキストセルの追加

メニュー
（目次，検索と置換，変数，ファイル）

コードセル，テキストセルの並び

Web ブラウザの画面

Google Colaboratory のノートブック

コードセル，**テキストセル**の２種類

- **コードセル**： Python プログラム，コマンド，実行結果
- **テキストセル**：説明文，図

２．式 ← テキストセル

```
[5] print(x + y)
    print(3 * x + y)
```

← コードセル

```
300
500
```

３．条件分岐 ← テキストセル

```
if (x > 50):
  print('big')
else:
  print('small')
```

← コードセル

```
big
```

演習３

ページ３４

【トピックス】

- **Google Colaboratory のアクセス**
- **コードセル、テキストセルの確認**

① Google Colaboratory のページを開く

https://colab.research.google.com/drive/1eQo91V3Tv4ldd0vJjdwJfmJs_MlbTw5m?usp=sharing

② プログラムや説明や実行結果が掲載されていることを確認

（このページは、次の演習４でも使用する）

2-4. Google アカウント

- Google アカウントは、無料で取得可能
- 各自で登録が必要
  - 氏名
  - メールアドレス
  - パスワード
  - 電話番号
  - 生年月日
  - 性別

- **各自でよく確認し、心配なことがある場合には取りやめてください。**

（その場合でも、授業の多くの部分は実施できるようにします）

Google アカウント

- **Google アカウント**は、Google のオンラインサービス等の利用のときに使うアカウント

  ID, パスワード

  （ID, パスワードは自分で決める）

- **Google Colaboratory** も利用

Google Colabotaroy と Google アカウント

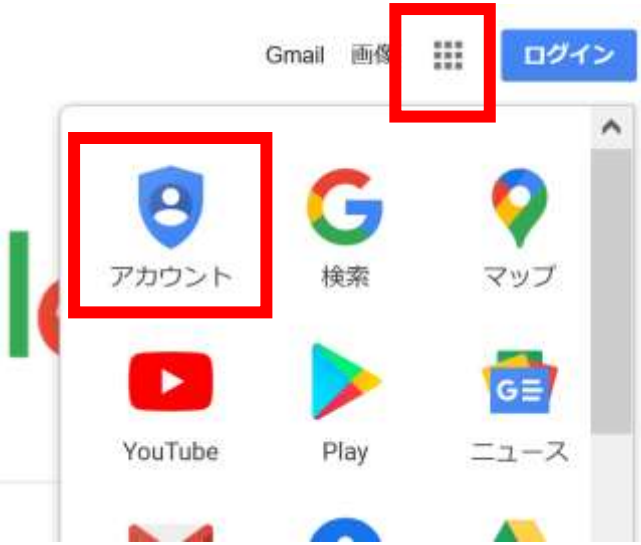
Google Colaboratory の本格的な利用には、Google アカウントが**必要**

- **ノートブック**の**新規作成，編集，保存，公開**
- **Python プログラム**の**編集，実行**
- **システム操作のコマンド**の**実行**
- **ファイル**の**アップロード，ダウンロード**

Google アカウントが**無くても**、他の人が公開しているノートブックの閲覧のみは可能

① **Google の Web ページ**を開く

https://www.google.com/



② 右上の**メニュー**で「**アカウント**」を選ぶ

③ 「**アカウントを作成する**」をクリック

④ 右のような画面が出たときは「**自分用**」を選ぶ

※「ビジネスの管理用」は Google の有料サービス G Suite のこと（関係ない）

⑤ 次の情報を登録する

姓，名

自分が希望するメールアドレス

＜ユーザー名＞ @gmail.com

パスワード（２か所），

電話番号

生年月日，性別

電話番号は
80-1234-5678
90-1234-5670
のように入れる
（先頭の 0 無し）

⑥ **本人確認のための電話番号**を入れ，「**次へ**」をクリック.

※ 電話番号は，スマートフォンの電話番号が便利

⑦その後、電話での指示に従う

全体まとめ

Google アカウントの取得

- Google アカウントは無料で取得可能。
- 必要な登録情報：氏名、メールアドレス、パスワード、電話番号（本人確認用）、生年月日、性別。
- Google のオンラインサービス（例：Google Colaboratory）で利用可能。

注意点

- 必要な情報をよく確認する。
- 心配な点があれば、取得を取りやめる。

手順

- Google の Web ページ https://www.google.com/
- 右上のメニューで「アカウント」を選ぶ。
- 「アカウントを作成する」をクリック。
- 「自分用」を選ぶ（「ビジネスの管理用」は有料サービス G Suite ）。
- 必要な情報を登録。
- 本人確認のための電話番号を入力。その後指示に従う。

2-5. Google Colaboratory の本格的な機能 （Google アカウントが必要）

Google Colaboratory の本格的な機能（使用には Google アカウントが必要）

- **ノートブック**の**新規作成，編集，保存，公開**（Google Drive との連携による）
- 公開により，**第三者がノートブックをダウンロードし，編集や実行なども可能**
- **Python プログラム**（コードセル内）の**編集，実行**
- 「!pip」や「%cd」などのシステム操作のための**コマンド**（コードセル内）の**編集，実行**
- **ファイル**の**アップロード**，**ダウンロード**
- ドキュメントの編集（図，リンク，添付ファイルを含めることができる）

コードセルは編集、実行可能

実行

```
files = ['a.png', 'b.png','c.png', '126.png', '127.png']
```

6．顔検出

顔検出は，写真やビデオの中の顔を検出すること．顔とそれ以外のオブジェクトを区別することも行う．顔検出の結果は，バウンディングボックスで得られるのが普通である．

次のプログラムは，Dlib を用いて，画像からの顔検出を行う。

- 「dets = cnn_face_detector(img, 6)」・・・顔検出の実行
- 「cv2.rectangle(disp, (d.rect.left(), d.rect.top()), (d.rect.right(), d.rect.bottom()), (255, 0, 0), 1)」・・・顔検出の結果を四角形で表示

結果は，赤い四角で表示される．1，3，4，5番目の画像（a.png, c.png, 126.png, 127.png）からは，顔が検出される．2番目の画像（b.png, 手で顔を覆い隠したもの）からは顔が検出されない．少し隠れていたり，顔が傾いていても顔検出ができるが，大きく隠れていると顔検出できない．

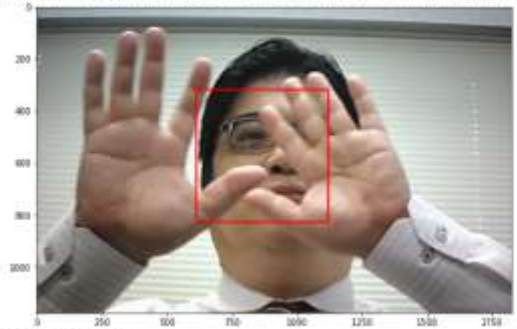
実行結果が長いので，スクロールして全体を確認すること．

謝辞: ここの Python プログラムは，Dlib に付属の cnn_face_detector.py を書き換えて使用している

実行

```
import sys
import dlib
import os
import urllib.request
import cv2
import matplotlib.pyplot as plt

cnn_face_detector = dlib.cnn_face_detection_model_v1('mmod_human_face_detector.dat')

for f in files:
    print("*** file: {} ***".format(f))
    img = dlib.load_rgb_image(f)
    dets = cnn_face_detector(img, 1)
    print("Number of faces detected: {}".format(len(dets)))
    for i, d in enumerate(dets):
        print("Detection {}: Left: {} Top: {} Right: {} Bottom: {} Confidence: {}".format(
            i, d.rect.left(), d.rect.top(), d.rect.right(), d.rect.bottom(), d.confidence))
    disp = img.copy()
    for i, d in enumerate(dets):
        cv2.rectangle(disp, (d.rect.left(), d.rect.top()), (d.rect.right(), d.rect.bottom()), (255, 0, 0), 9)
    plt.figure(figsize=(10,10))
    plt.imshow(disp)
    plt.show()
```

```
*** file: a.png ***
Number of faces detected: 1
Detection 0: Left: 614 Top: 318 Right: 1121 Bottom: 827 Confidence: 0.2080110013464648
```



```
*** file: b.png ***
```

コードセル

テキストセル

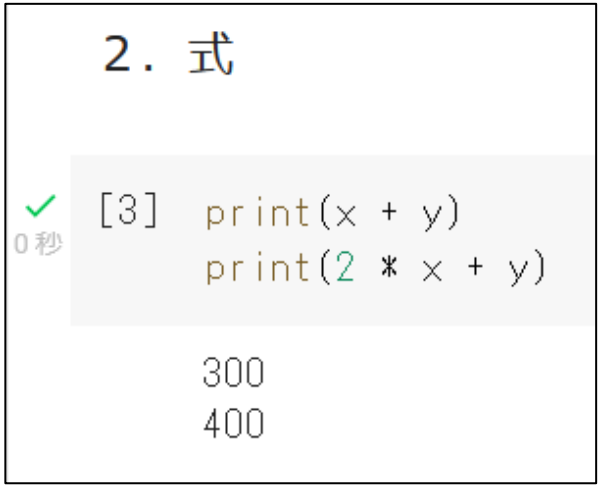
コードセル

- **WEBブラウザでアクセス**
- **コードセル**は **Python プログラム**．
  各自の Google アカウントでログインすれば，
  変更，再実行可能

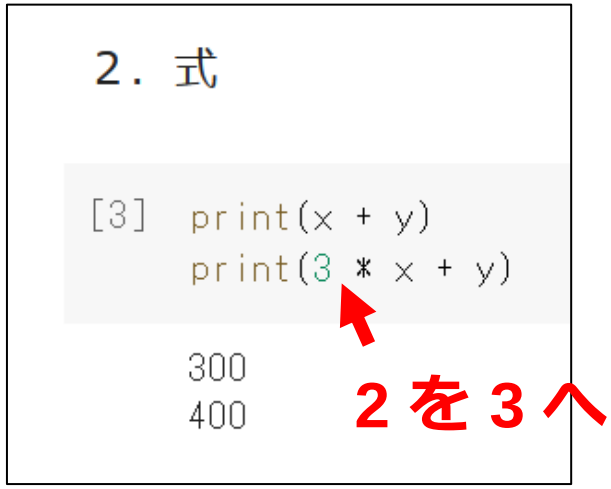
一番上のコードセルから順々に実行

コードセルとプログラム実行

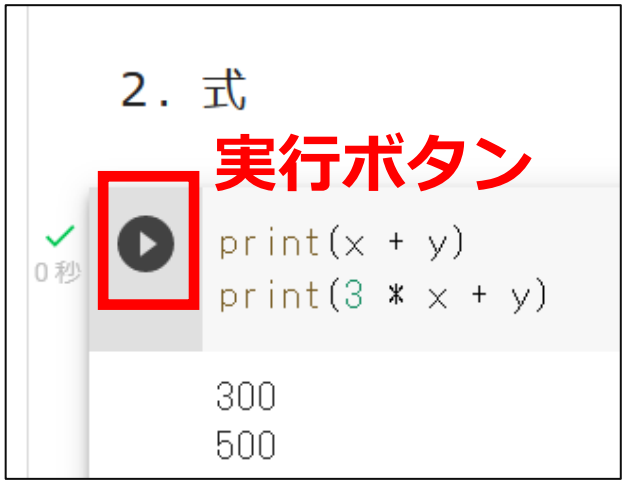
コードセルで，
Python プログラムやコマンドの編集，実行ができる．
（編集や実行には **Google アカウントが必要**）



編集前



編集後



実行ボタンと
実行結果

Google Colaboratory でうまく実行できない場合

混雑しているときなどは、**実行が止まり、再開しない場合**もある

【その対処】

次で、**アクティブなセッションの停止**を行い、その後最初から実行をやり直す

- **メニュー**で「**ランタイム**」，「**セッションの管理**」と操作する.
- **アクティブなセッションの一覧**が表示されるので，「**終了**」をクリックして，**すべてのアクティブなセッションを終了**する.

Google Colaboratoryの要点

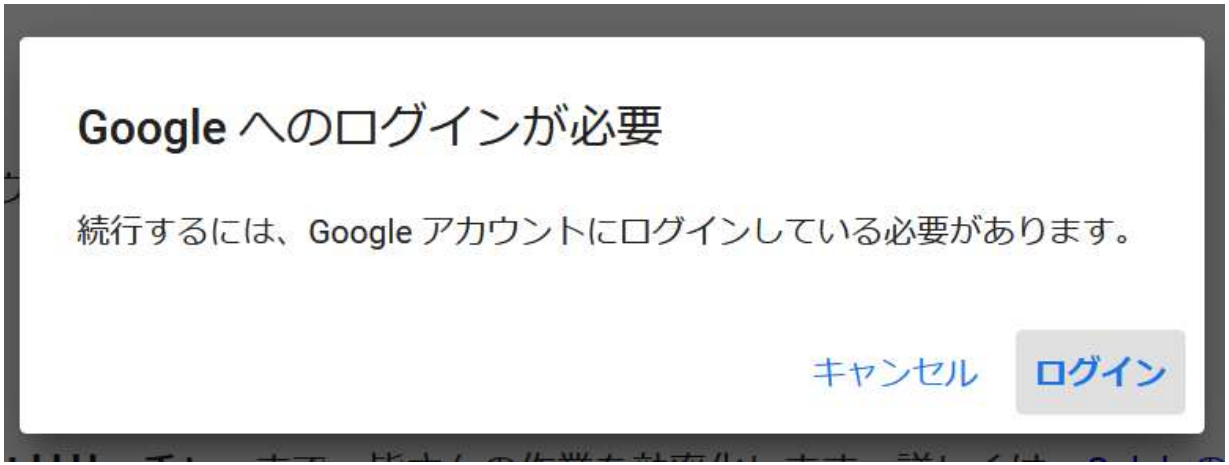
- **アクセス**: Webブラウザからアクセス可能。
- **セルの種類**: コードセル（プログラム用）、テキストセル（説明用）。

基本操作

- **Googleアカウント**: 基本操作には**Googleアカウントが必要**。
- **操作の種類**: コードセルやテキストセルの**編集**、セルの**実行**、新規ノートブックの作成など。
- **保存**: 自動保存される。
- **セルの実行**: **基本、一番上のセルからすべてを実行してください**。このような、**複数のセルを一度**に実行することは、「**ランタイム**」メニューから「**すべてのセルを実行**」の操作でできます。

Google アカウントでのログイン

Google Colaboratoryの**使用中**に
「Google へのログインが必要」と表示されたとき
Google アカウントへのログインを行う。



まとめ①

Google Colaboratoryの基本

- オンラインで動作するPythonのノートブック環境
- 本格的な利用には、Googleアカウントが必要
- Pythonと多くのライブラリがプリインストールされている

主な機能

- ノートブックの新規作成、編集、保存、公開（Google Driveと連携）
- Pythonプログラム（コードセル内）の編集、実行
- システム操作コマンド（!pip, %cdなど）の実行
- ファイルのアップロード、ダウンロード
- ドキュメントの編集（図、リンク、添付ファイルを含む）

まとめ②

ノートブックの構造

- コードセルとテキストセルの2種類がある
- コードセルでPythonプログラムやコマンドの編集、実行
- テキストセルで説明文や図を表示

トラブルシューティング

- 混雑しているときは実行が止まる場合がある
- 「ランタイム」→「セッションの管理」でアクティブなセッションを終了して再開

Googleアカウントについて

- ログインが必要な操作：ノートブックの新規作成、編集、保存、公開、プログラムの実行など
- ログインが不要な操作：他人が公開したノートブックの閲覧

2-6. Python の Pandas データフレーム

Python の Pandas データフレーム

表形式のデータ

	x	y
0	1	4
1	1	2
2	1	5
3	2	4
4	3	5
5	3	3

データ本体

Python の Pandas データフレームを組み立てるプログラム

	x	y
0	1	4
1	1	2
2	1	5
3	2	4
4	3	5
5	3	3

データフレーム

```
import pandas as pd

df = pd.DataFrame([[1, 4], [1, 2], [1, 5], [2, 4], [3, 5], [3, 3]], columns=['x', 'y'])
print(df)
```

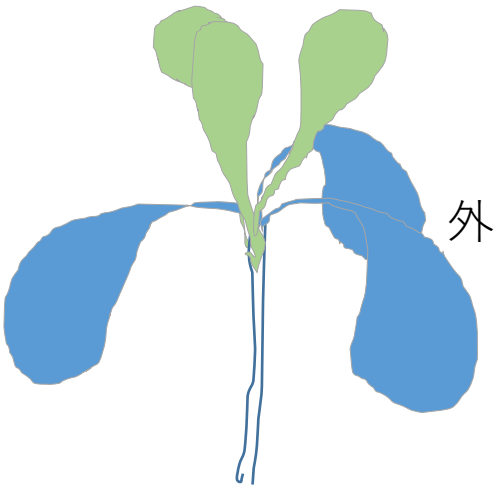
```
   x  y
0  1  4
1  1  2
2  1  5
3  2  4
4  3  5
5  3  3
```

データフレームの作成と確認表示

データフレームの組み立て，結果は **df** に代入

```
df = pd.DataFrame([[1, 4], [1, 2], [1, 5], [2, 4], [3, 5], [3, 3]], columns=['x', 'y'])
```

アヤメ属 (Iris)



- 多年草
- 世界に 150種. 日本に 9種.
- 花被片は 6個
- **外花被片**（がいかひへん） Sepal
  3個（大型で下に垂れる）
- **内花被片**（ないかひへん） Petal
  3個（直立する）

Pandas データフレームの重要性

数多くの機能を持つ

【主な機能】

- **データ分析**
- **データクリーニング**: 欠損データの処理や、データの変換・正規化
- **時系列データ**: 時間属性を含むデータを扱う機能
- **データの統合**: データの結合、マージ、グループ化

【柔軟性】

様々なデータ形式（CSV, Excel, リレーショナルデータベース等）をサポート。

【性能】

大規模なデータセットも効率的に処理できる

Iris データセット

Iris データセット（データ数は 50 × 3）
のうち、先頭 10 行

sepal_length	sepal_width	petal_length	petal_width	species
5.1	3.5	1.4	0.2	setosa
4.9	3	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa
4.6	3.4	1.4	0.3	setosa
5	3.4	1.5	0.2	setosa
4.4	2.9	1.4	0.2	setosa

外花被片(**Sepal**)の長さと幅　**内花被片**(**Petal**)の長さと幅　**種類**

- **3種**のアヤメの**外花被辺**、**内花被片**を計測
- 種類も記録

  setosa

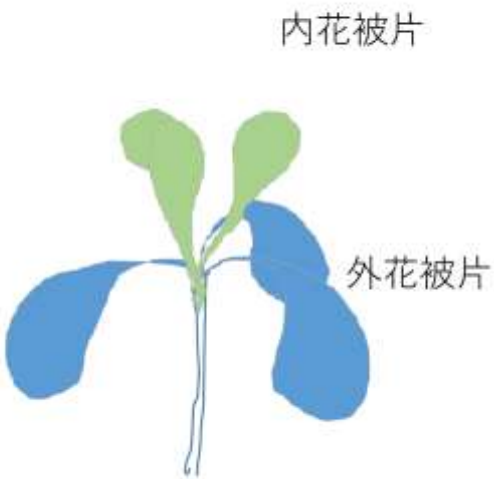
  versicolor

  virginica
- データ数は **50** × **3**

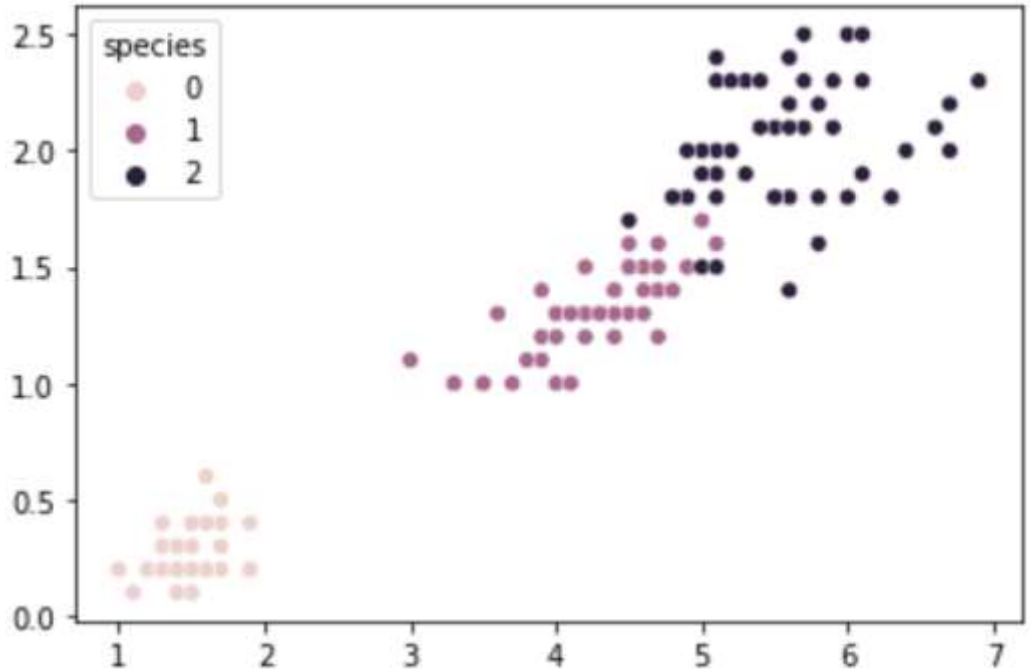
作成者：Ronald Fisher

作成年：1936

Iris データセットの散布図



縦軸：内花被片の**幅**



横軸：内花被片の**長さ**

次の**3種類**の分類済みのデータ

0: **setosa**
1: **versicolor**
2: **virginica**

2-7. Google Colaboratoryを活用してデータを取り扱う実践的な演習

Pandas の基本的な使い方

- Google ColaboratoryにはpandasがＰ**既にインストール**されている。
- **インポート**: pandasを使用する前に、import pandas as pdというコードでインポート。

<u>主要なメソッド</u>

- **CSVファイルの読み込み**: read_csvメソッド。

  例: pd.read_csv('file.csv')

- **Excelファイルの読み込み**: read_excelメソッド。

  例: pd.read_excel('file.xlsx')

- **データの先頭部分、末尾部分を表示**: head(), tail() メソッド。

  例: df.head() df.tail()

- **１列の選択**: [ ]を使用して列を選択。

  例: df['column_name']

- **複数列の選択:** [[ ]]を使用して列を選択。

  例: df[['a1', 'a2']]

データ分析の基本手順: 読み込みから散布図の作成まで

データの準備と確認

- **データの準備**: CSVファイルやExcelファイルを読み込むことも可能。

  例: df = pd.read_csv('file.csv') または df = pd.read_excel('file.xlsx')

- **データの先頭部分、末尾部分を表示（データが大きい場合）**: head(), tail() メソッド。

  例: df.head() df.tail()

列の選択（データに多数の列がある場合）

- [ ] （１列の選択）や [[ ]] （１列あるいは複数列の選択）を使用して列を選択。

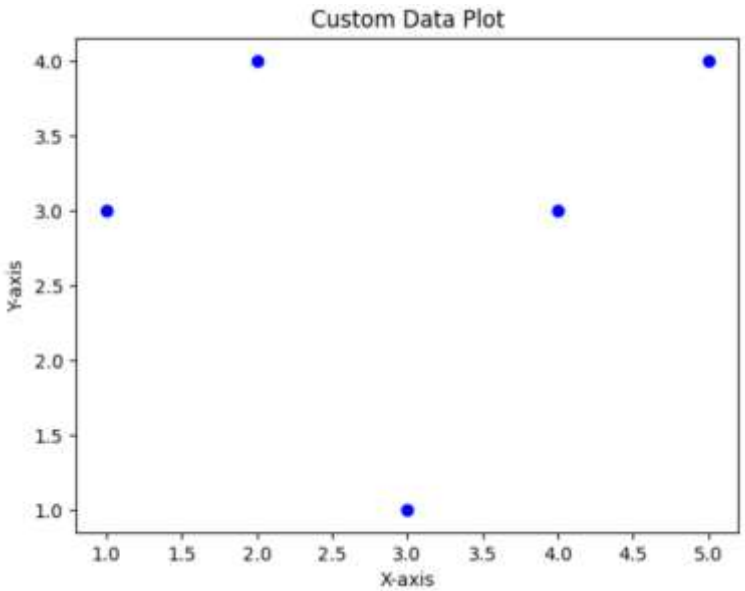
  例: df[‘column_name’] df[['a1', 'a2']]

散布図の作成

- **Matplotlibのインポート**: 例 import matplotlib.pyplot as plt
- **２列から散布図を作成**: plt.scatter を使用
- **軸ラベルの追加:** 例 plt.xlabel('X-axis label') と plt.ylabel('Y-axis label')
- **タイトルの追加**: 例 plt.title('Scatter Plot Title')
- **最後にグラフを表示**: plt.show()

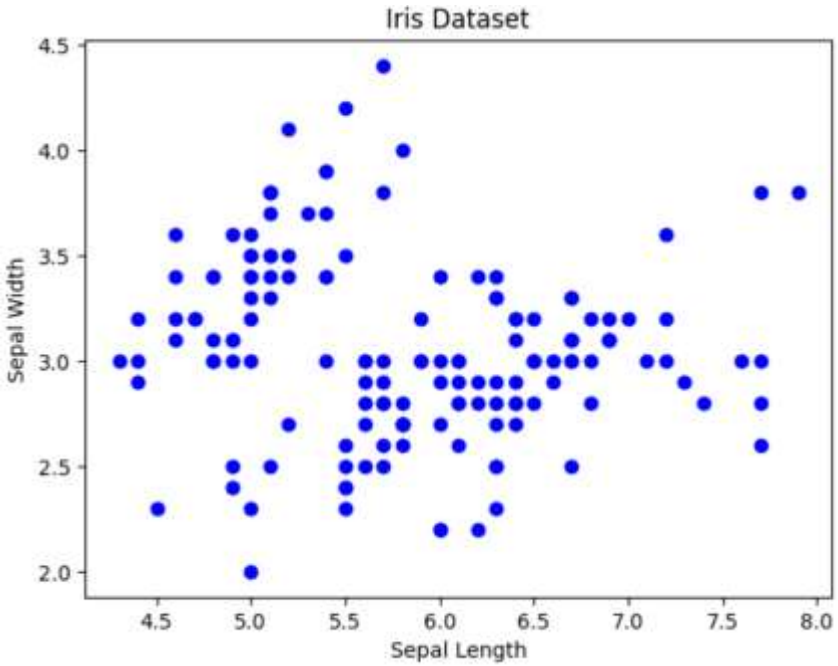
①簡単な散布図

```python
# 必要なライブラリをインポート
from sklearn.datasets import load_iris
import matplotlib.pyplot as plt
import pandas as pd

# 新しい X と y の値
X = [1, 2, 3, 4, 5]
y = [3, 4, 1, 3, 4]

# Matplotlibを用いて結果をプロット
plt.scatter(X, y, color='blue')

# 軸ラベルとタイトルを追加
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Custom Data Plot')

# グラフを表示
plt.show()
```



② Iris データセットのロードと散布図（Pandas データフレームを使用）

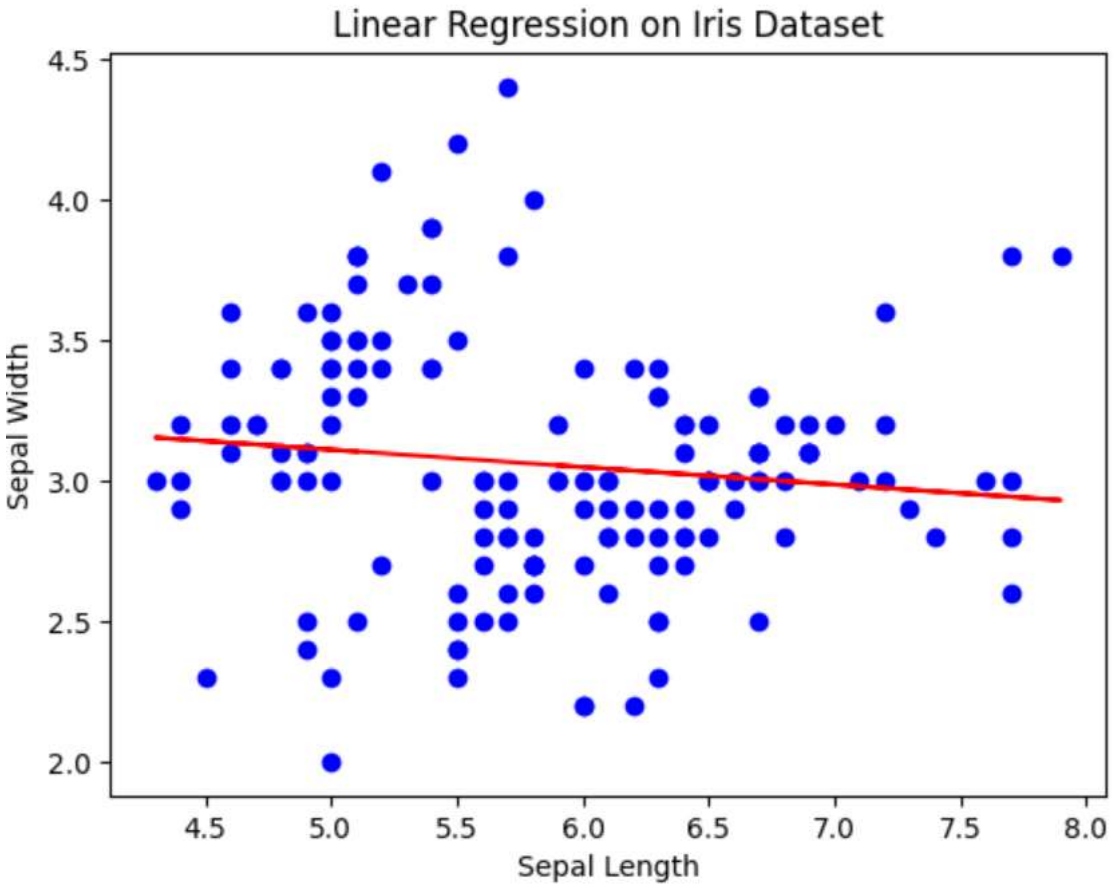
```python
# 必要なライブラリをインポート
from sklearn.datasets import load_iris
import matplotlib.pyplot as plt
import pandas as pd

# irisデータセットをロード（pandas を使用）
iris = load_iris()
df = pd.DataFrame(iris.data, columns=iris.feature_names)

# データの先頭と末尾を確認
print("データの先頭:")
print(df.head())
print("データの末尾:")
print(df.tail())

# 必要な列だけを選択（DataFrame形式でスライス）
X = df[['sepal length (cm)']]
y = df['sepal width (cm)']

# Matplotlibを用いて結果をプロット
plt.scatter(X, y, color='blue')

# 軸ラベルとタイトルを追加
plt.xlabel('Sepal Length')
plt.ylabel('Sepal Width')
plt.title('Iris Dataset')

# グラフを表示
plt.show()
```



③ 機械学習の例（回帰）

- 機械学習のうち１つ「**回帰**」を行う。ここでは、**データに最もよく適合する線を求めている**

```python
# 必要なライブラリをインポート
from sklearn.linear_model import LinearRegression
from sklearn.datasets import load_iris
import matplotlib.pyplot as plt
import pandas as pd

# irisデータセットをロード (pandas を使用)
iris = load_iris()
df = pd.DataFrame(iris.data, columns=iris.feature_names)

# データの先頭と末尾を確認
print("データの先頭:")
print(df.head())
print("データの末尾:")
print(df.tail())

# 必要な列だけを選択 (DataFrame形式でスライス)
X = df[['sepal length (cm)']]
y = df['sepal width (cm)']

# 線形回帰モデル
model = LinearRegression()

# 学習
model.fit(X, y)

# 予測を行う
y_pred = model.predict(X)

# Matplotlibを用いて結果をプロット
plt.scatter(X, y, color='blue')
plt.plot(X, y_pred, color='red')

# 軸ラベルとタイトルを追加
plt.xlabel('Sepal Length')
plt.ylabel('Sepal Width')
plt.title('Linear Regression on Iris Dataset')

# グラフを表示
plt.show()
```



演習４

ページ６４

【トピックス】

- **データ分析の基本手順: 読み込みから散布図の作成まで**
- **Google Colaboratory の本格的な機能**

① Google Colaboratory のページを開く（演習 3 と同じ）

② プログラムと説明をよく読み、掲載されている実行結果をよく確認して、データ分析の基本手順を、各自で読み取る

自習

- **１つ目の「簡単な散布図」関する自習**
- 点の色を赤に変更してください。

ヒント: １つ目のコードセルについて、plt.scatter()のcolor引数を ‘blue‘ から **’red’に変更**してみてください。そして実行して確認してください。

（提出する必要はありません）

自習

- **２つ目の「Iris データセットのロードと散布図」関する自習**
- Iris データセットで、今回使わなかった次の２つの列

  petal length (cm)

  petal width (cm)

  を使うようにプログラムを書き換えてください
- ヒント: ２つ目のコードセルについて、次の変更を行って再実行してください

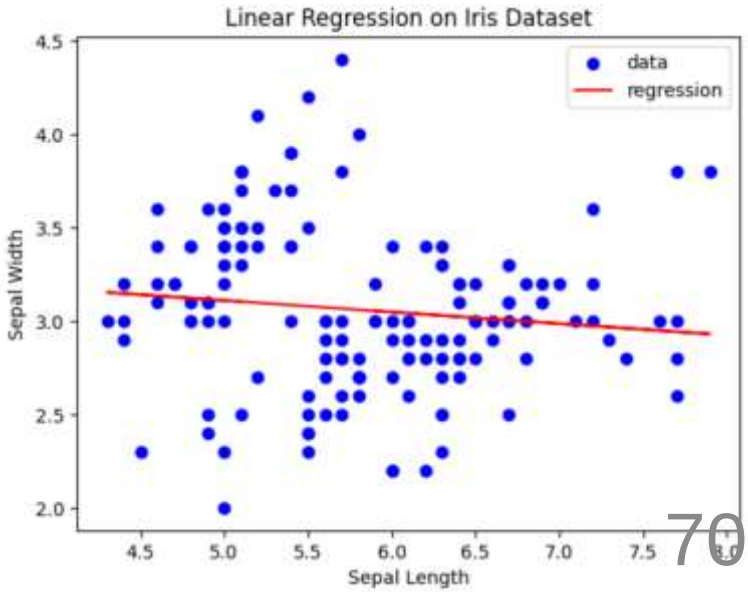
```
X = df[['petal length (cm)']]
y = df['petal width (cm)']
```

（提出する必要はありません）

自習

- **３つ目の「機械学習の例（回帰）」関する自習**
- Python の Matplotlib を用いたグラフ作成について、授業では教えなかった「凡例」について、自習。青い点が実際のデータ、緑の線が予測線であることを示してください。
- ヒント

```
# Matplotlibを用いて結果をプロット
plt.scatter(X, y, color='blue', label='data')
plt.plot(X, y_pred, color='red', label='regression')

# 軸ラベルとタイトルを追加
plt.xlabel('Sepal Length')
plt.ylabel('Sepal Width')
plt.legend()
plt.title('Linear Regression on Iris Dataset')
```



全体まとめ①

Pythonの特徴

- Pythonは**広く利用されているプログラミング言語**の1つ。
- 特徴として、読みやすさ、書きやすさ、幅広い応用範囲。

プログラミングの良さ

- **プログラミングは人間の力を増幅**し、多くの活動で役立つ。
- シミュレーション、データ処理、AI、ITシステム制作などで利用。
- プログラミングは**クリエイティブな行為**。

変数、代入、オブジェクト、メソッド

- **変数**は**オブジェクトに名前**を付けたもの。値を保存し、後から参照できる。
- **代入**操作を通じて**変数に値を保存**。
- オブジェクトは操作の対象であり、**メソッドはそのオブジェクトが持つ機能**を表す。
- メソッドは**引数**を指定して呼び出す。

Pythonの文法と基本的な機能

- Pythonはシンプルで読みやすい文法。
- 機能として変数、データ型、条件分岐、繰り返し、関数、オブジェクト指向プログラミング、モジュール、例外処理などがある。

全体まとめ②

Pythonのコーディングスタイル

- **字下げ**を使用して**ブロックの範囲**を示す。
- **空白行**や**コメント**を使い、**コードを読みやすく**する。

Google Colaboratory

- **オンラインで動作**し、Pythonの**ノートブック機能**を提供する。
- **Googleアカウント**により、**コードセル**や**テキストセル**で**プログラムを編集・実行**できる。

PythonのPandasデータフレーム

- **Python で表形式のデータを扱う**ためのライブラリ。
- CSVやExcelファイルの読み込み、データ選択、散布図の作成に利用。

データ分析の基本手順

- データの準備と確認。
- 列の選択（データに多数の列がある場合）。
- 散布図の作成。

2-8. 追加の自習
（提出の必要なし）

自習

これは、Google Colaboratory の
ノートブックを作成する演習です。
自習とします。

【トピックス】

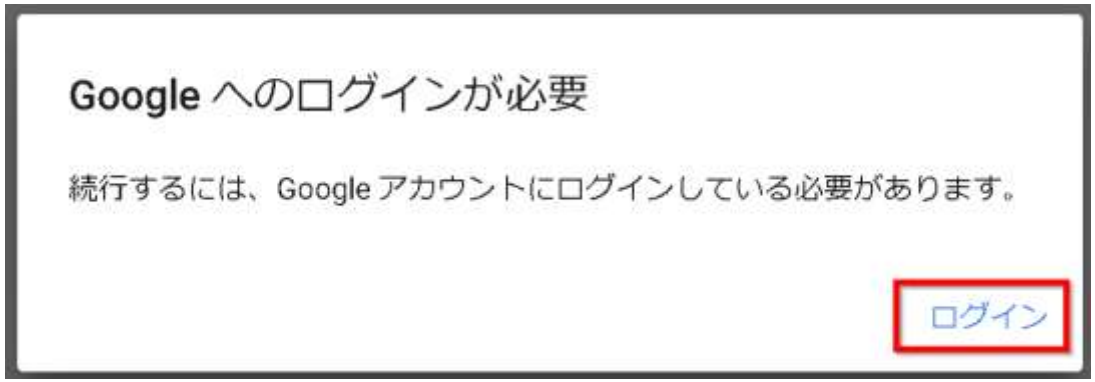
- Google Colaboratory
- コードセルの作成
- Python プログラムの実行

① Google Colaboratory のWebページを開く

https://colab.research.google.com

② 「**ファイル**」で，「**ノートブックを新規作成**」を選ぶ

③ **Google アカウント**でのログインが求められたときは**ログイン**する

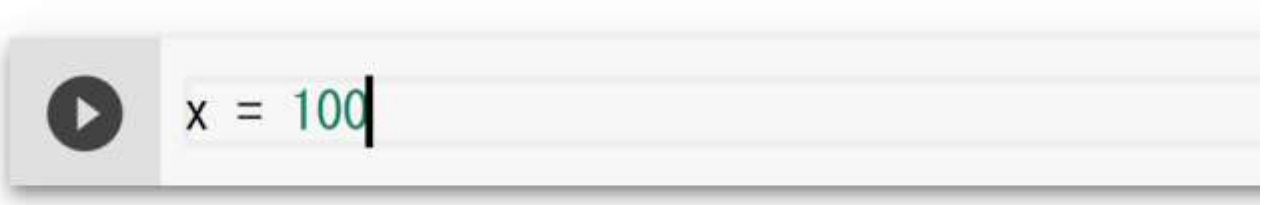


④ コードセルの新規作成

メニューの「**+ コード**」をクリック

⑤ コードセルに Python プログラムを入れる

x = 100



⑥ コードセルの新規作成

メニューの「**+ コード**」をクリック

⑦ コードセルに Python プログラムを入れる

```
if x > 20:
  print("big")
else:
  print("small")
```

if x > 20 の直後に「:」
else の直後に「:」
（どちらも，コロン）

字下げ．
（ここでは，半角の空白を 2つ）

⑧ コードセルの新規作成

メニューの「**+ コード**」をクリック

⑨ コードセルに Python プログラムを入れる

```
s = 0
for i in [1, 2, 3, 4, 5]:
  s = s + i
print(s)
```

for i in [1, 2, 3, 4, 5] の直後に「:」

字下げ.
（ここでは，半角の空白を 2つ）

⑩ コードセルを上から順に実行し，結果を確認

```
[4] x = 100
```

コードセルの実行ボタンをクリック

```
if (x > 20):
  print("big")
else:
  print("small")
```

```
big
```

コードセルの実行ボタンをクリック

```
s = 0
for i in [1, 2, 3, 4, 5]:
  s = s + i
print(s)
```

```
15
```

コードセルの実行ボタンをクリック