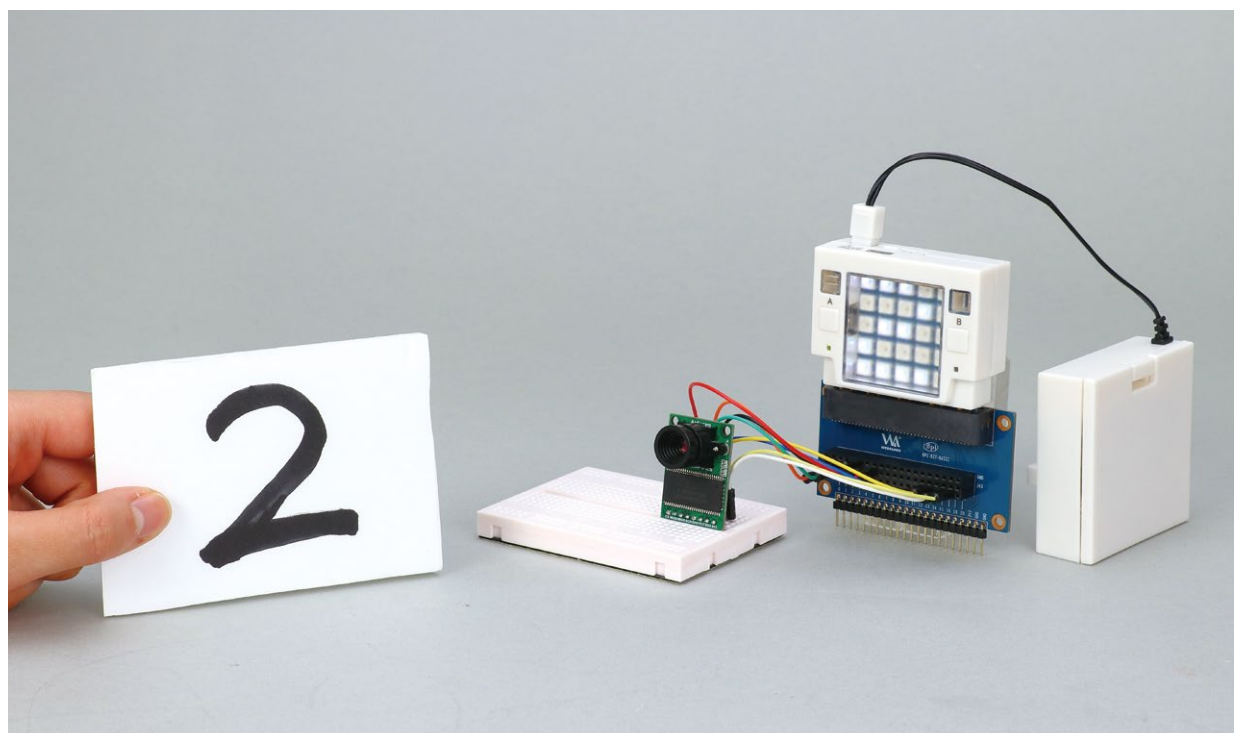


高校情報Ⅱキット 手書き数字認識システム (AI+IoT)

教材監修:元 東京都立日比谷高等学校教諭／元 東京学芸大学特任教授 天良和男先生

教員用



目次

学習テーマ1 環境の準備 3

準備

準備① Pythonの開発環境	4
準備② MicroPython開発環境	11
準備③ 小型カメラの用意	17

学習テーマ2 Webサーバの構築とカメラの制御 18

演習

演習① Webサーバの構築	19
シェバンについて	25
Webサーバが起動しない場合のトラブルシューティング	27
演習② 小型カメラの制御	31
wifi_config関数について	37
wifi_ifconfig関数について	37

学習テーマ3 AIを使った手書き数字認識 46

準備

準備④ 人工知能(AI)とはなにか	47
準備⑤ 手書き数字認識システム動作の流れ	52
準備⑥ ライブラリのインストール	53

演習

演習③ scikit-learnのデータセットを確認	55
演習④ 画像の前処理	61
演習⑤ 教師データの学習と手書き数字の予測	72
演習⑥ 手書き数字認識システムの構築	75

付 録

付録 A. ブレッドボードの使い方	81
付録 B. 変換エッジコネクタの使い方	81
付録 C. display オブジェクトの機能一覧	82
付録 D. Wi-Fi機能の一覧	83
付録 E. モバイルホットスポットの使い方	84
付録 F. scikit-learnの手書き数字のデータセット	88
付録 G. 機能学習モデルの比較	90
付録 H. cgiモジュールを使用しないプログラム	91
画像データに関して	93

学習テーマ 1

環境の準備

[学習内容]

- Python の開発環境
- MicroPython 開発環境
- 小型カメラの用意

本テキストに掲載しているプログラムは下記ページからダウンロードできます。

<https://www.artec-kk.co.jp/dl/93275/>

準備① Python の開発環境

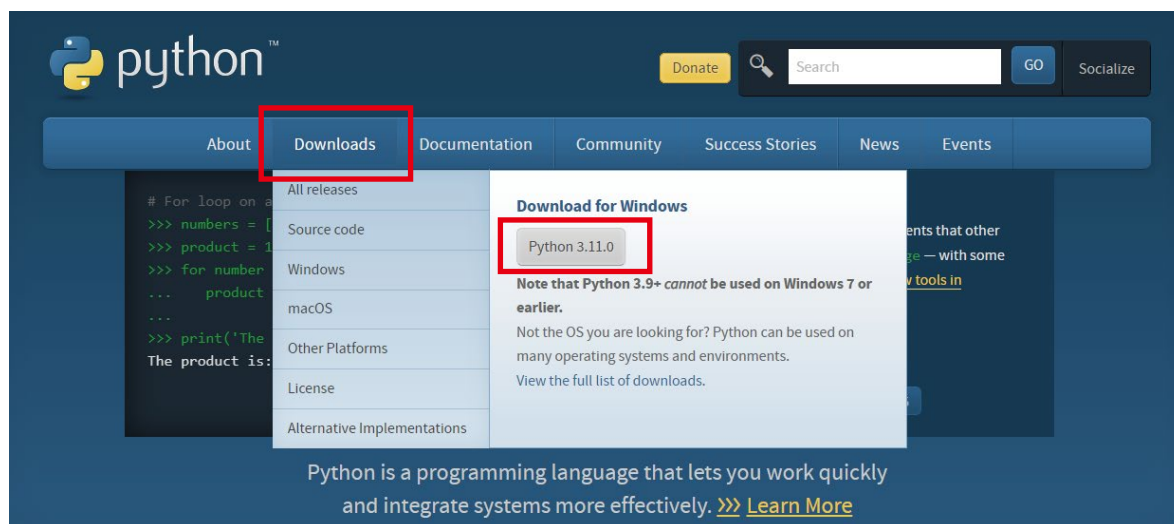
1. Python のインストール

サーバとして使用する PC において、以下より Python をダウンロードしてインストールしてください。

<https://www.python.org/>

※以下の手順は 2022 年 12 月現在のものになり、予告なく変更される場合があります。

① Downloads を選択し、使用する OS にあったバージョンをダウンロードします。



※Pythonのバージョンとcgiモジュールの注意事項

演習⑤、演習⑥ではPythonの標準モジュールであるcgiモジュールを使用します。

Python3.11やPython3.12でcgiモジュールを使ったプログラムを実行するとプログラムは正常に動作しますが、警告メッセージが表示される可能性があります。

Python3.13以降のバージョンでは正常に動作しない可能性が考えられます。

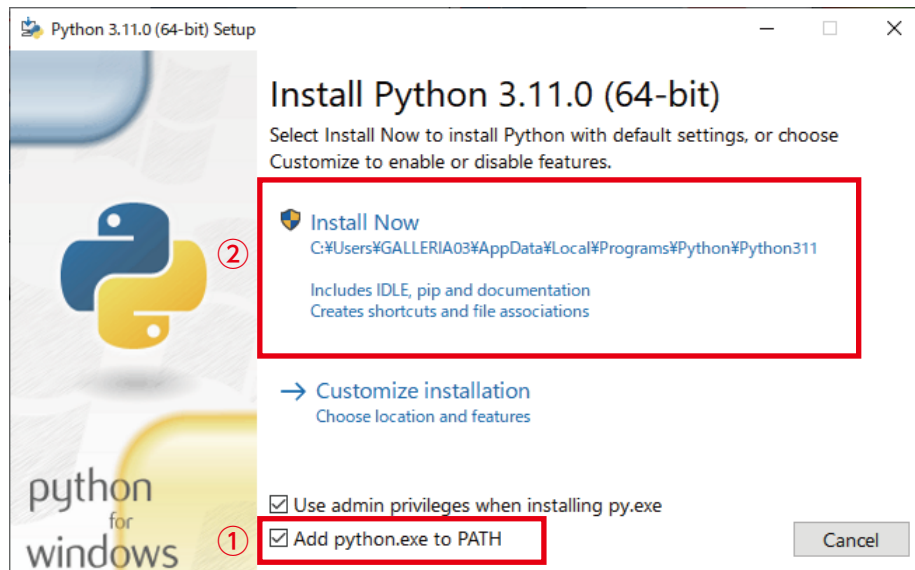
Windows環境でPython3.13以降のバージョンを使用する場合は、付録Gを参照してcgiモジュールを使わないプログラムを作成してください。

Mac環境ではcgiモジュールを使わない方法には対応していないため、Python3.12以前のバージョンをインストールして、cgiモジュールを使ったプログラムを作成してください。

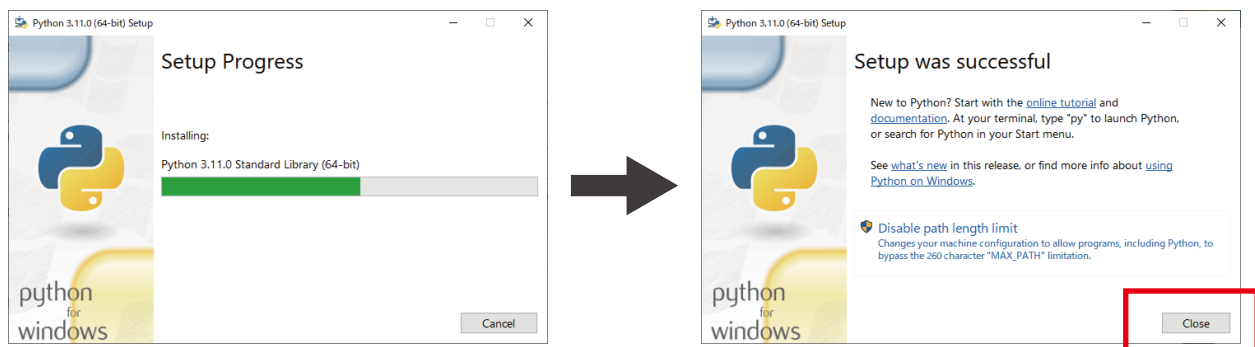
詳細は付録Hを参照してください。

②ダウンロードされた「python-3.*.*.exe」をダブルクリックします。

立ち上がった画面で「Install launcher for all users(recommended)」と「Add python.exe to PATH」に☑を入れてから、「Install Now」をクリックします。



③「Setup was successful」が表示されたら「Close」をクリックしてインストールは完了です。



※すでにPythonをインストール済みのPCを使用する場合の注意事項

- PythonのWebサーバ機能を使用する際、②のパスの設定(☑)の有無によって実行方法が異なります。
- 本テキストに記載のserver.pyのサンプルプログラムはPython3.7以降のバージョン用に作成されています。Python3.6以前のバージョンでは正常に動作しません。

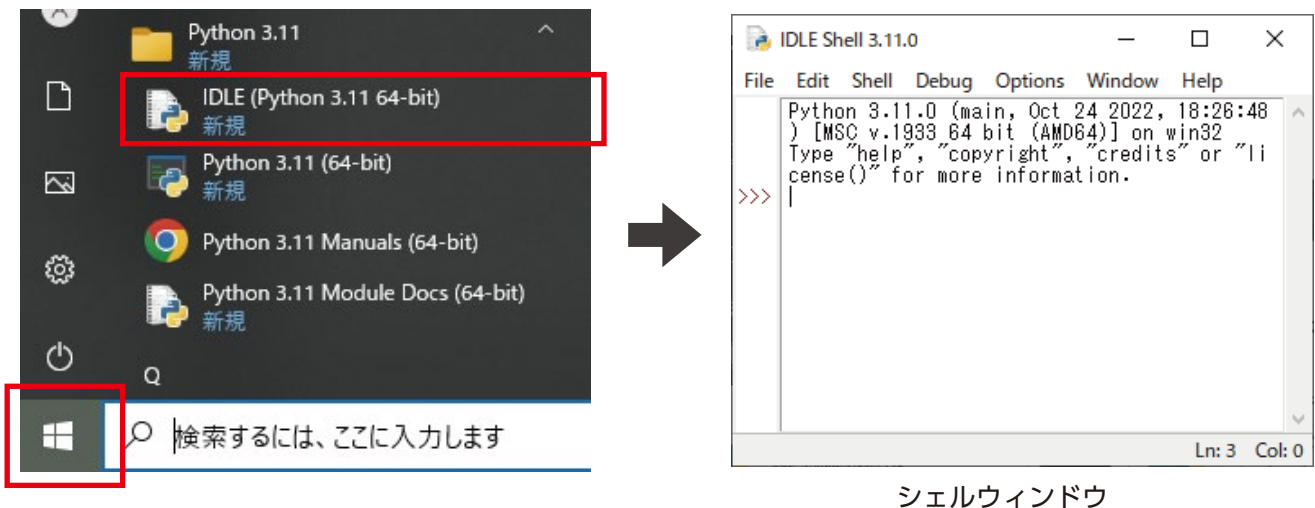
2. IDLE(統合開発環境) の使い方

Pythonをインストールした際に一緒にインストールされている[IDLE]を使用する手順を下記に記します。

IDLEにはエディタ (編集)、プログラムの実行、デバッグなどの機能があります。

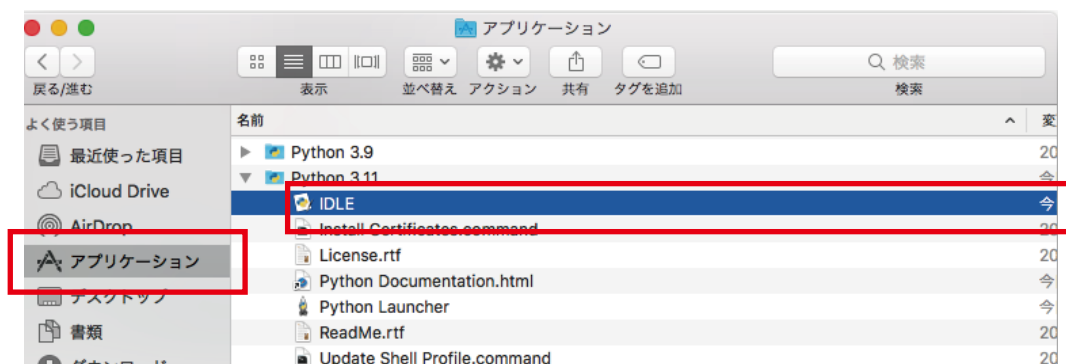
Windows 環境

PCのスタートメニューからPythonフォルダ内のIDLE を選択すると、IDLEが起動しシェルウィンドウが表示されます。



Mac 環境

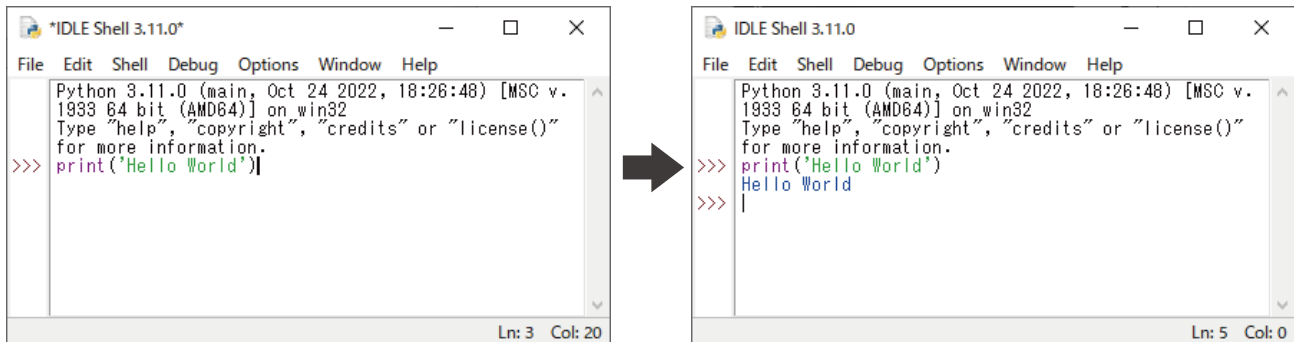
Finderを起動して「アプリケーション」から使用したいバージョンの「Python」フォルダ内にあるIDLEを選択すると、IDLEが起動しシェルウィンドウが表示されます。



2-1. 対話モードで実行する

シェルウィンドウの「>>>」(プロンプト)のあとにコードを入力すると、すぐに実行結果を確認できます。ただし、1行ずつしか実行できません。

`print('Hello World')` を実行して、「Hello World」が表示されることを確認してください。

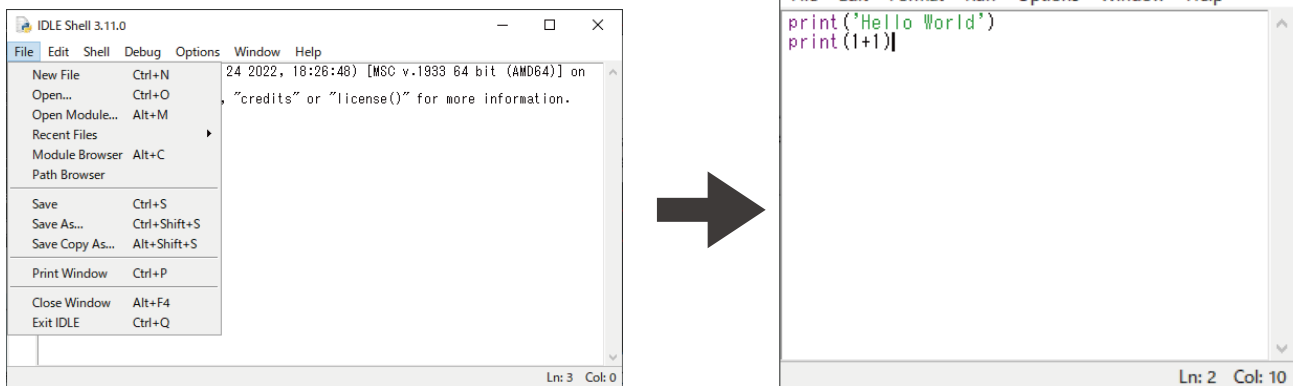


2-2. 新規ファイルを作成し、ファイル実行モードで実行する

①「File」、「New File」の順にクリックするとエディタウィンドウが開きます。

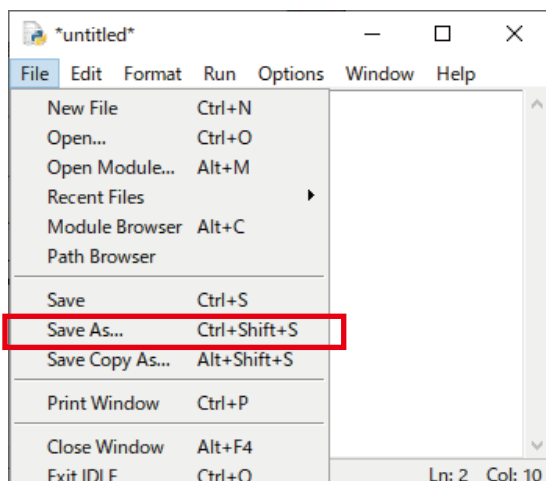
下記のプログラムを入力してください。

```
print('Hello World')
print(1+1)
```

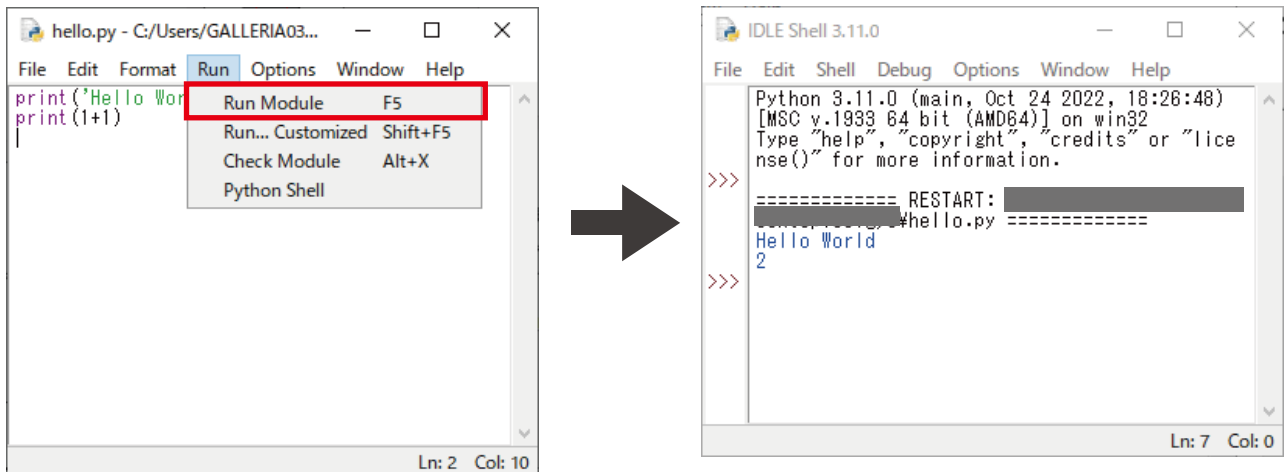


エディタウィンドウ

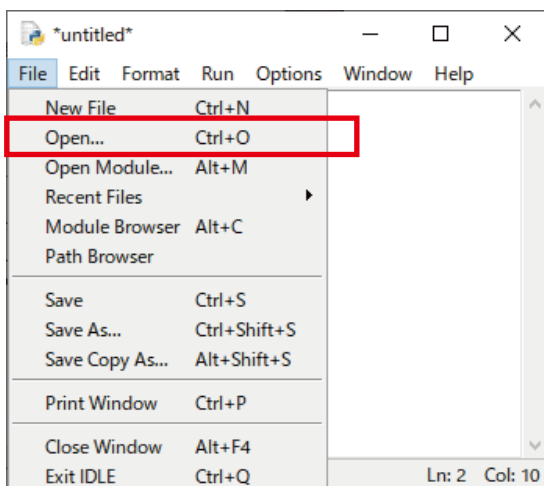
②「File」、「Save As」の順にクリックしてファイルに名前を付けて保存します。



③エディタウィンドウの「Run」、「Run Module」の順にクリックすると、実行結果がシェルウィンドウに表示されます。



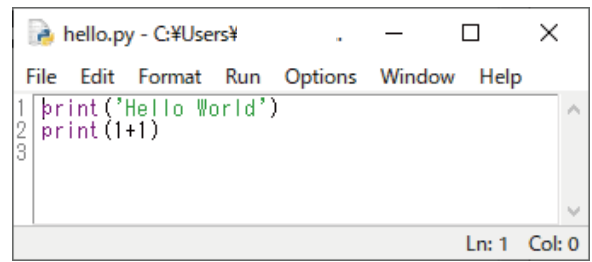
④保存したファイルを開く場合は「File」、「Open」の順にクリックしてください。



2-3. 表示の設定変更

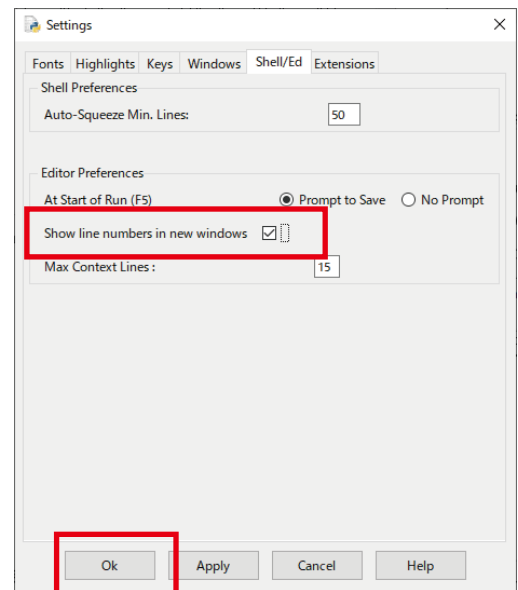
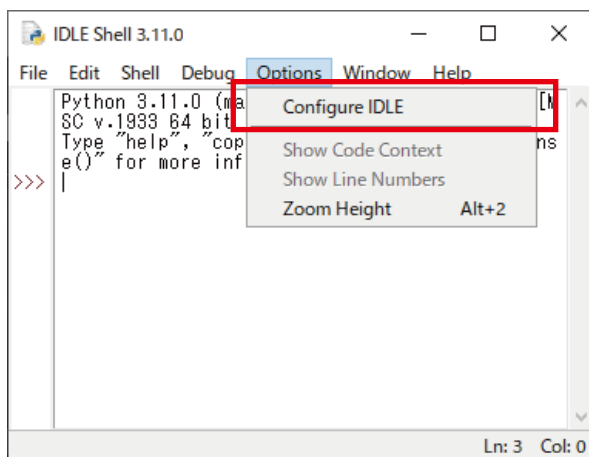
●行番号の表示

Python3.8 以降の場合、行番号を表示することが可能です。



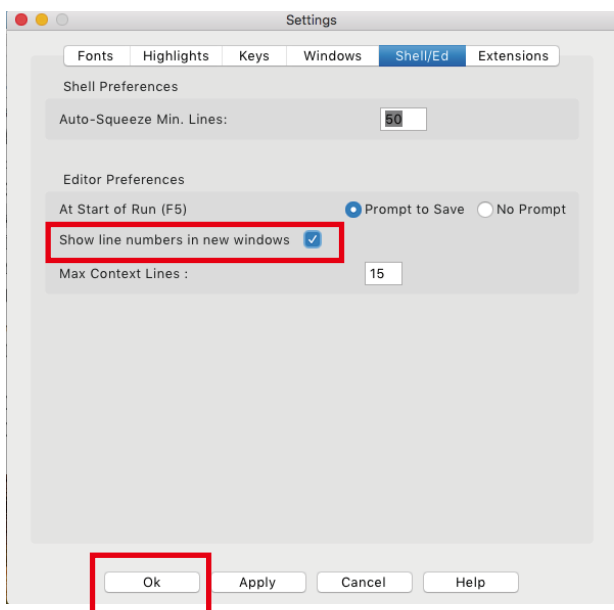
Windows 環境

シェルウィンドウまたはエディタウィンドウから「Option」、「Configure IDLE」、「Shell/Ed (または General) タブ」の順にクリックし、「Show line numbers in new windows」をチェックして「OK」をクリックすると、行番号が表示されます。



Mac 環境

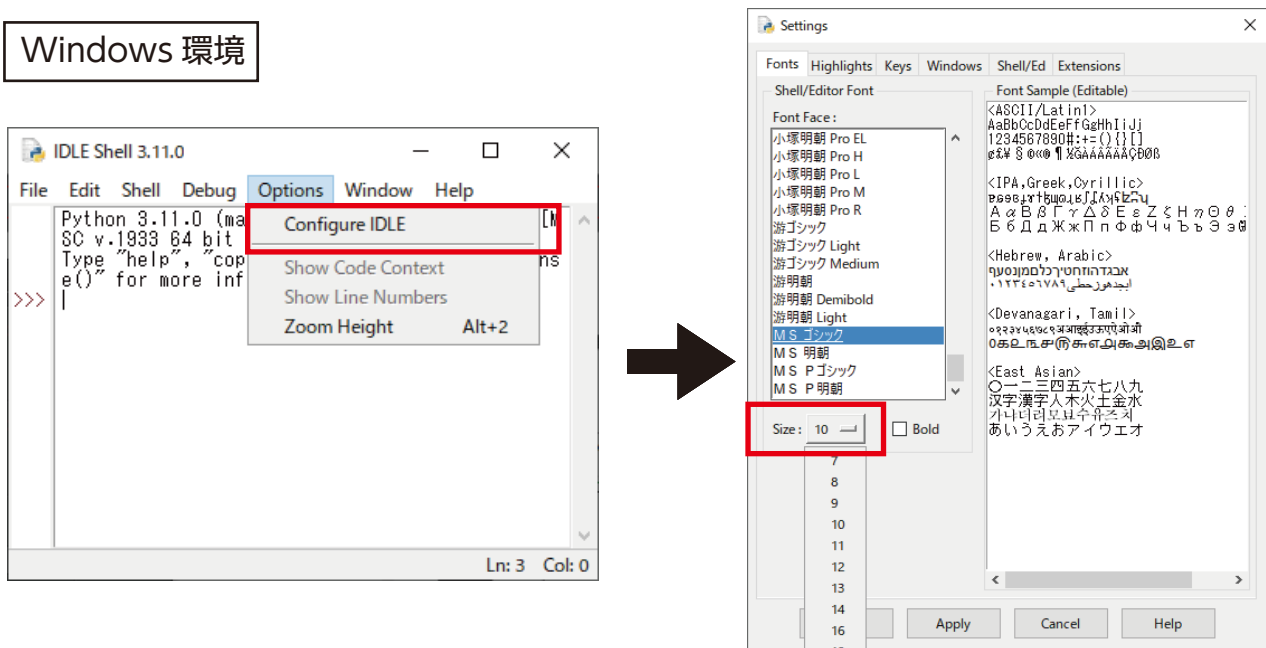
シェルウィンドウまたはエディタウィンドウから「IDLE」、「Preferences」、「Shell/Ed (または General) タブ」の順にクリックし、「Show line numbers in new windows」をチェックして「OK」をクリックすると、行番号が表示されます。



●文字サイズの変更

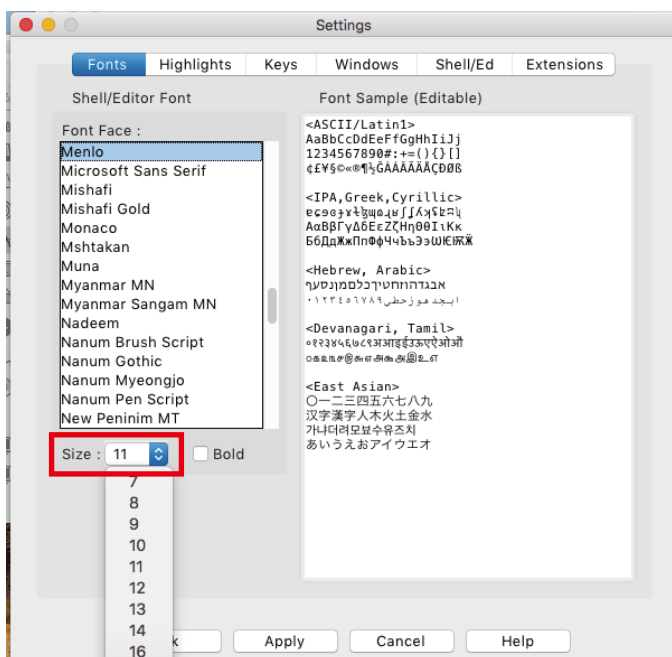
シェルウィンドウまたはエディタウィンドウから「Option」、「Configure IDLE」の順にクリックし、「Size:」横のボタンをクリックして文字サイズを選択できます。「OK」をクリックすると、選択した文字サイズに変更されます。

Windows 環境



Mac 環境

シェルウィンドウまたはエディタウィンドウから「IDLE」、「Preference」の順にクリックし、「Size:」横のボタンをクリックして文字サイズを選択できます。「OK」をクリックすると、選択した文字サイズに変更されます。



準備② MicroPython 開発環境

1. ソフトウェアのインストール

Studuino:bitなどのボード型コンピュータで動作するPythonは、普通のPythonではなく、MicroPythonです。MicroPythonはStuduino:bitなどのボード型コンピュータで実行できるように実装されたPython言語の処理系で、Python3.0との間で高い互換性を有します。MicroPythonを使ったプログラムはMuエディタなどを使って作成します。

ここではStuduino:bit用に機能拡張されたMuエディタを使用します。

下記ArtecRobo2.0のPythonページ内にあるドキュメントを参考に、Studuino:bit用Muエディタをダウンロード、インストールしてください

<https://www.artec-kk.co.jp/artecrobo2/ja/software/python.php>

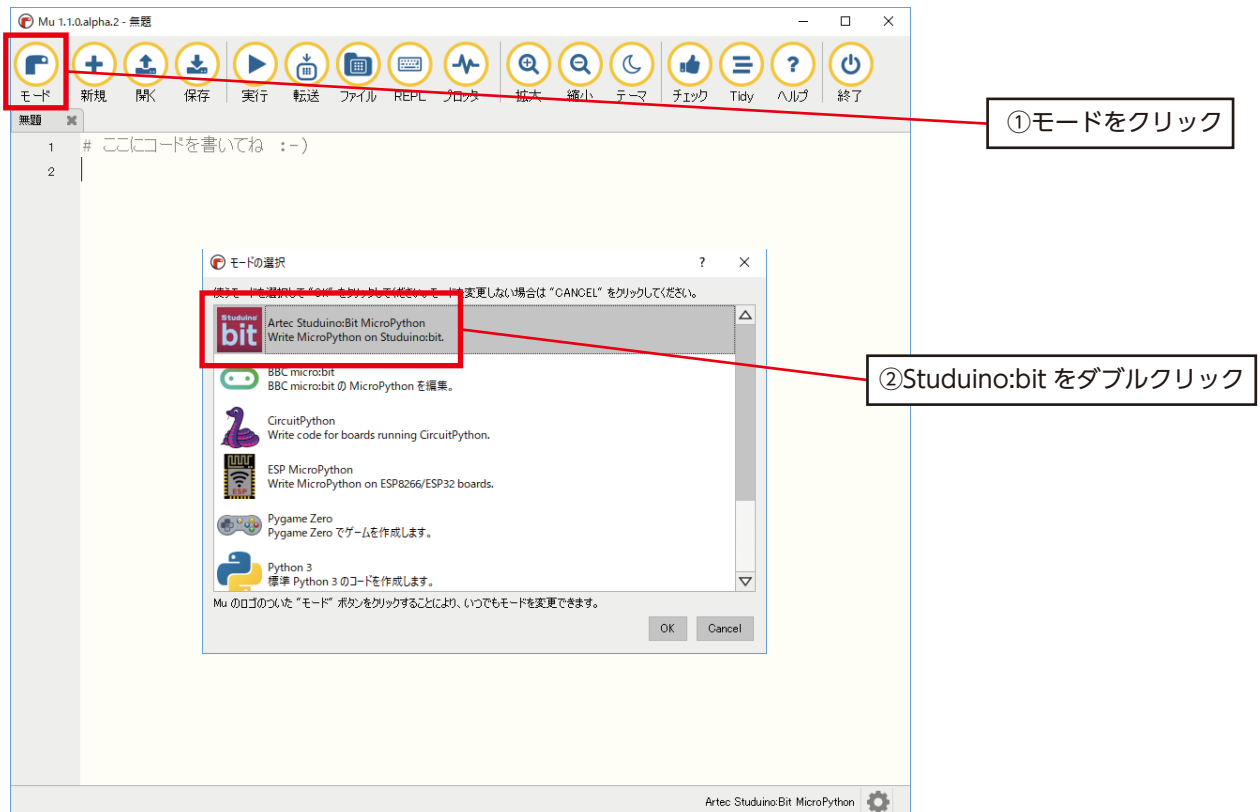
本テキストでの使い分け

Python	MicroPython
PC 上で動作する Web サーバの役割	Studuino:bit 上で動作する
IDLE でプログラムを作成 	Mu エディタでプログラムを作成 

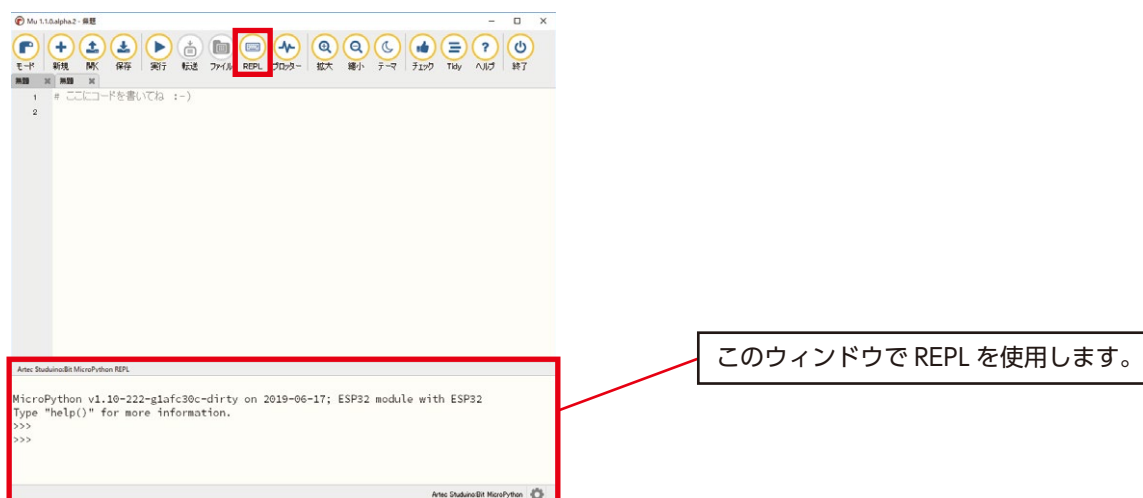
2.REPL でプログラムを実行する

Python の対話型インタプリタREPL (Read-Evaluate-Print Loop)を使用する手順を下記に記します。

- ① Studuino:bitとPCを接続します。
- ② Muを起動して、下図のように「モード」ボタンをクリックして、「Artec Studuino:bit MicroPython」をダブルクリックしてください。



- ③「REPL」ボタンをクリックすると、Muウィンドウの下にウィンドウが表示され、入力待ちを示す「>>>」が表示されます。REPL表示中は「転送」ボタン、「ファイル」ボタンは灰色表示になり使用できません。再度「REPL」ボタンをクリックすることでウィンドウが非表示になり、「転送」ボタン、「ファイル」ボタンを使用できるようになります。

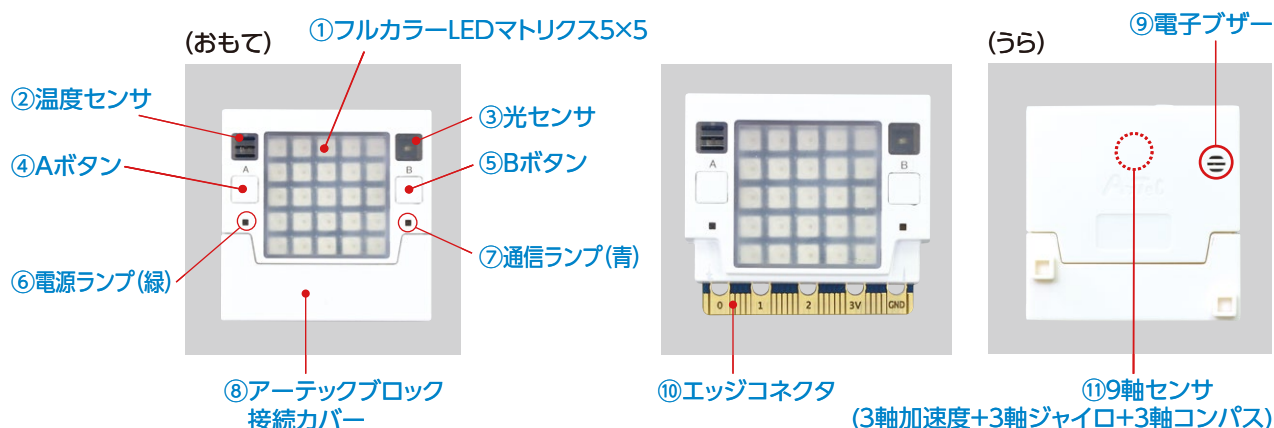


- ④REPLの自動エコーによって、ウィンドウの「>>>」の後にプログラムを入力すると、すぐに結果を見ることができます。REPLで下記を実行し、「Hello World」が表示されることを確認してください。

```
>>>print('Hello World')
```

3.Studuino:bit ライブラリ

Studuino:bitを制御する、オブジェクトや関数をまとめたモジュールとしてStuduino:bitライブラリを提供しています。Studuino:bitは下図のハードウェアを持っています。



Studuino:bitライブラリを使用することで、Studuino:bitのハードウェアを簡単に制御できます。ライブラリには、Studuino:bitの各ハードウェアに対応したオブジェクトが定義されています。

番号	ハードウェア	オブジェクト	機能概要
①	フルカラーLEDマトリクス5×5	display	各LEDのON/OFFやLEDを使ったアニメーションなど
②	温度センサ	temperature	温度センサの値の取得
③	光センサ	lightsensor	光センサの値の取得
④	Aボタン	button_a	AボタンのON/OFF判定など
⑤	Bボタン	button_b	BボタンのON/OFF判定など
⑨	電子ブザー	buzzer	電子ブザーで鳴らす音の高さをON/OFFなど
⑩	エッジコネクタ	p0~p16,p19,p20	エッジコネクタの入出力制御
		pin0~pin3	micor:bit 互換エッジコネクタの入出力制御 (※)
⑪	3軸加速度センサ	accelerometer	加速度センサの値の取得
	3軸ジャイロセンサ	gyro	ジャイロセンサの値の取得
	3軸地磁気センサ	compass	地磁気センサの値の取得

※:pin0とp0、pin1とp1、pin2とp2、pin3とp3の併用はできません。

※:pin0~pin3オブジェクトは、p0~p3オブジェクトと同じメソッドが使用できますが、read_analog()メソッドは0~1023の値を返します。

各オブジェクトは、pystubit.boardモジュールに定義されています。REPLで下記を実行し、Studuino:bitメインユニットの画面に文字列”Hello, MicroPython”がスクロール表示されることを確認してください。

```
>>> from pystubit.board import display
>>> display.scroll("Hello, MicroPython")
```

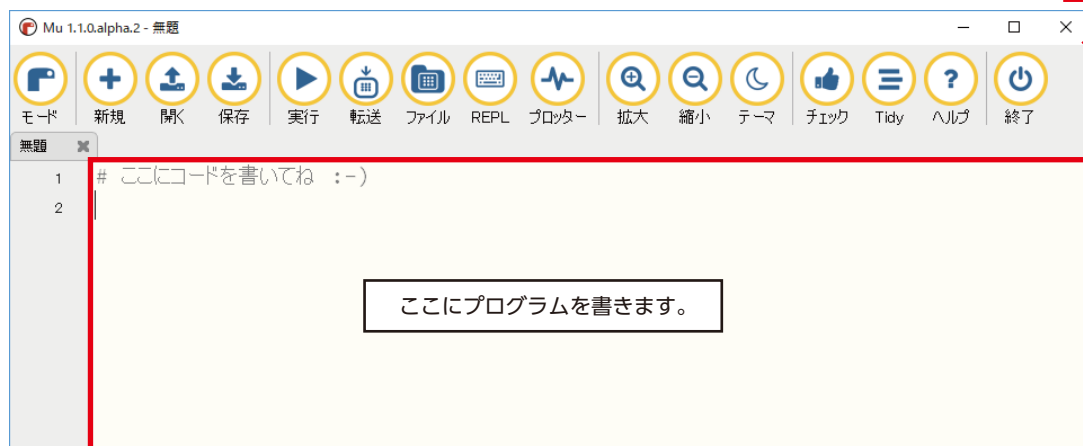
4. 「実行」 ボタンでプログラムを実行する

Muエディタで編集したPythonスクリプトをStuduino:bitで実行する手順を下記に記します。

① 下記をテキストエディタに入力します。

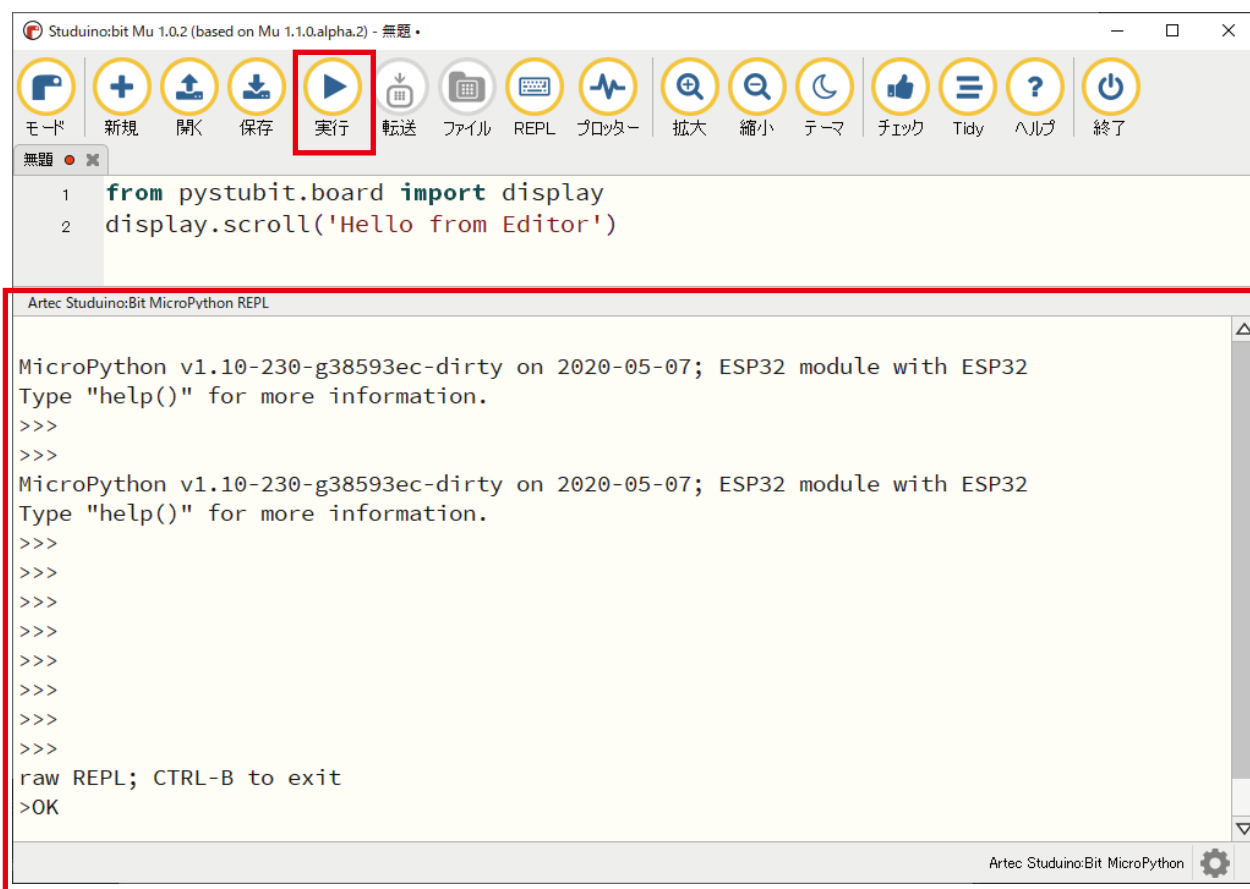
```
from pystubit.board import display
display.scroll('Hello from Editor')
```

テキストエディタ



② 「実行」 ボタンをクリックするとStuduino:bitでプログラムが実行されます。

Studuino:bitメインユニットの画面に文字列”Hello from Editor”がスクロール表示されることを確認してください。



「実行」 ボタンをクリックすると下側にウィンドウが表示され、REPLを使用できるようになります。

5.「転送」ボタンでプログラムを実行する

Muエディタで編集したPythonスクリプトをStuduino:bitに転送し、実行する手順を下記に記します。

① 4.①のプログラムをテキストエディタに入力し、「転送」ボタンをクリックします。

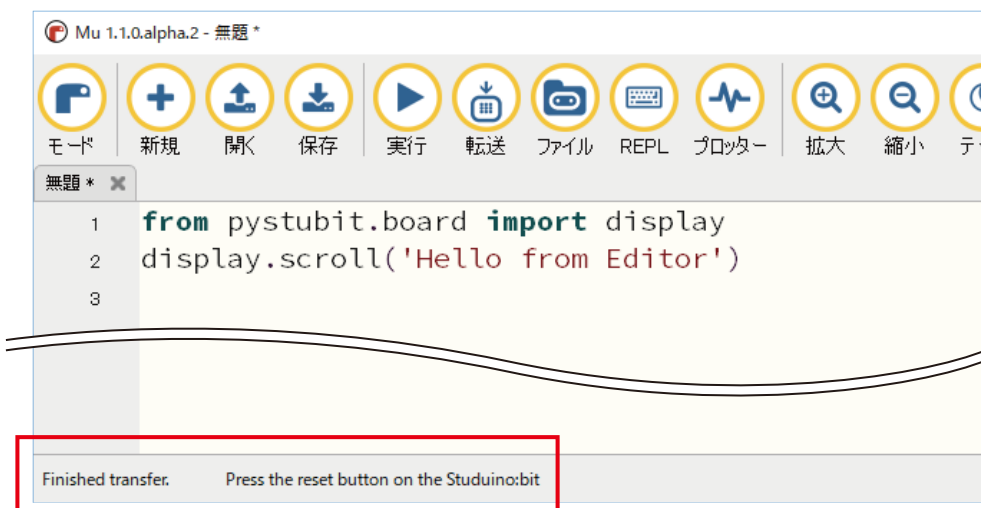
※「転送」ボタンが無効(灰色表示)になっている場合は「REPL」ボタンをクリックし、REPLウィンドウを閉じることで「転送」ボタンが有効になります。



②表示されるダイアログで、プログラムの転送先0～9のいずれかの「Transfer」ボタンをクリックします。



ステータスバーに「Finished transfer.」と表示されたら、転送完了です。

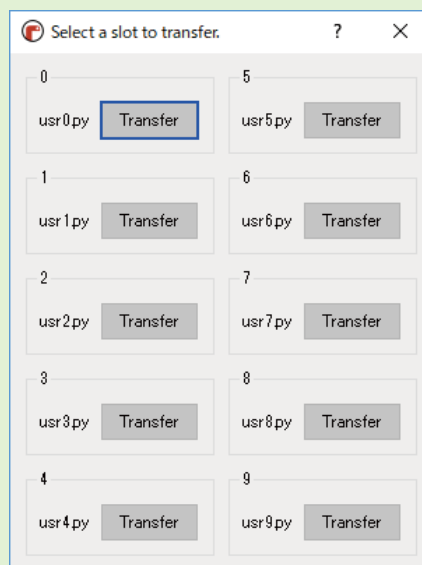


Studuino:bitメインユニットの画面に文字列”Hello from Editor”がスクロール表示されることを確認してください。表示終了後は、Studuino:bitメインユニットのリセットボタンを押すことで再度プログラムが実行されます。プログラムは転送されているため、USB ケーブルを取り外し、電池ボックスを接続してスタンドアローンで動作を確認できます。

●複数のプログラムを転送した場合の実行方法

複数のプログラムを0～9の番号を指定して転送することができます。

通常は最後に転送したプログラムが実行されますが、以下の方法で番号を指定して転送済みのプログラムを実行することができます。



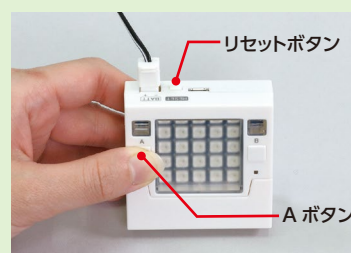
①電源を接続する(通常起動)

- (1) 電池ボックスもしくはUSBケーブルをメインユニットに接続します。
- (2) 電源ランプ(緑)が点灯し、電源が自動的にONになり起動します。通常起動時は前に実行したプログラムが自動的に実行されます。



②プログラム選択モードへの移行

- (1) Aボタンを押したまま、リセットボタンを押してください。リセットボタンを押すと再起動のため一時的に電源ランプが消灯します。
- (2) リセットボタンを離して3秒後に電源ランプが再点灯したら、Aボタンを離します。
- (3) 緑のLEDで0が表示されるとプログラム選択モードへ移行しています。



③プログラムの選択・起動

- (1) プログラム選択モード時にAボタンを押すと、表示が0・1・2...と切り替わります。9まで切り替わると0に戻ります。
- (2) 選択した番号でBボタンを押すと、各番号に割り振られたプログラムが実行されます。次回より通常起動時に(2)で選択したプログラムが自動で実行されます。

※工場出荷段階で動作確認用に各番号にはサンプルプログラムが書き込まれています。

※サンプルプログラムはソフトウェアを使用して新たなプログラムを書き込むと上書きされます。



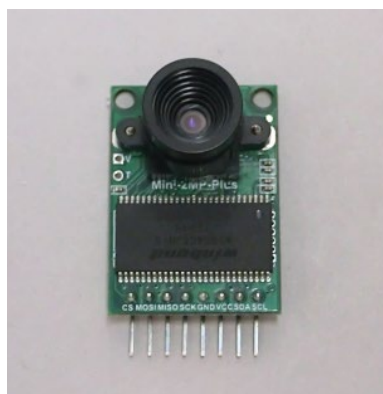
準備③ 小型カメラの用意

このテキストでは小型カメラをStuduino:bitに接続して、手書きの数字が書かれた紙を撮影します。

使用する小型カメラはUCTRONICSのArducam Miniモジュール (OV2640) です。

本製品のセットには含まれていないため別途ご購入下さい。

※本テキストで提供する撮影用のライブラリは上記小型カメラに対応したものですので、他の小型カメラでは動作しません。



製品情報はこちら

<https://www.uctronics.com/>

ご購入方法が不明な場合は、アーテックまでお問い合わせください。

学習テーマ 2

Web サーバの構築とカメラの制御

[学習内容]

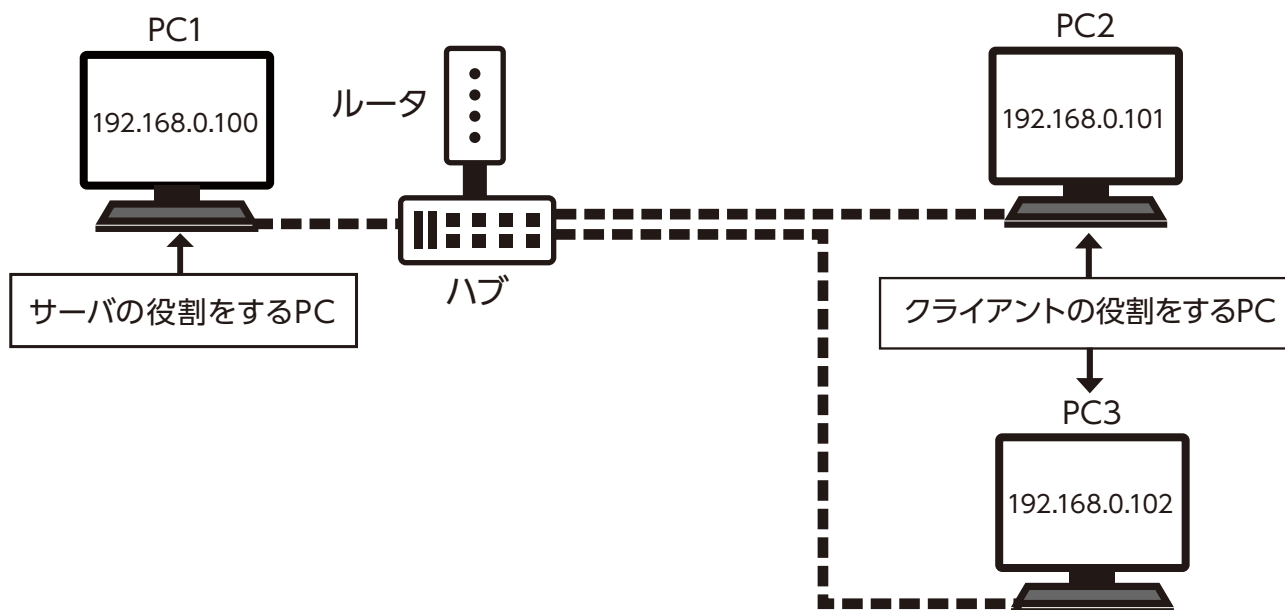
- Web サーバの構築
- 小型カメラの制御

演習① Webサーバの構築

Pythonには標準モジュールとしてhttp.serverというWebサーバの機能があります。ここでは、これを利用してコンピュータ教室などのLAN(ローカルエリアネットワーク)内の任意のPCをWebサーバとして動作させ、同じLAN内のPCをクライアントとして動作させます。そのための設定手順を以下に記載します。

図中のルータやハブ、PC1、PC2、PC3 は、コンピュータ教室に設置されている既存の装置を使用します。

ネットワーク環境例 ※IP アドレスの数値は一例

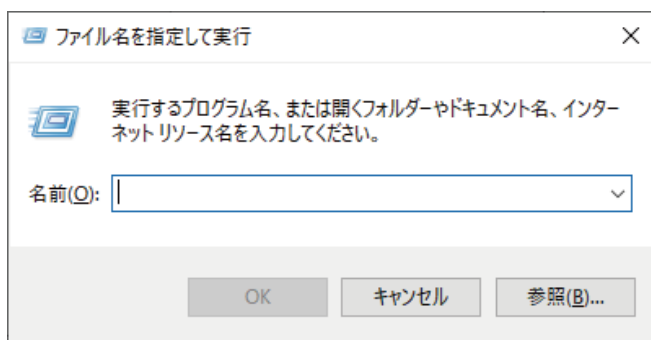


1.IP アドレスの確認

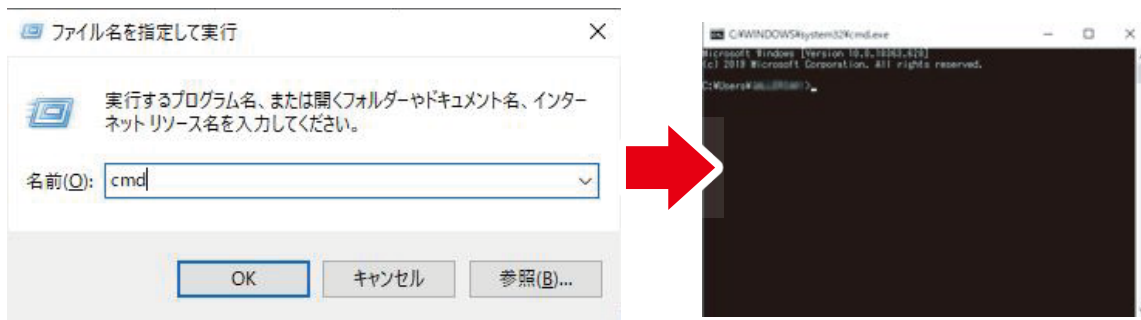
サーバの役割をする PC に割り当てられている IP アドレスを確認します。

Windows 環境

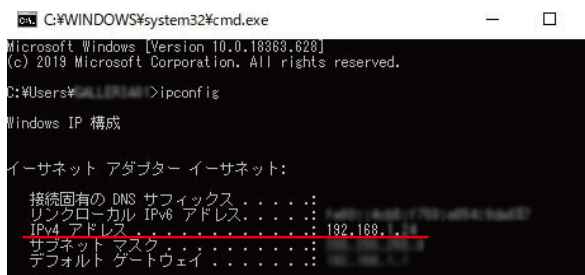
- ① 「Windows キー」+R キーを押してください。画面左下に下記のようなウィンドウが表示されます。



- ② 「cmd」と入力し、「OK」ボタンをクリックすると、コマンドプロンプトが起動します。

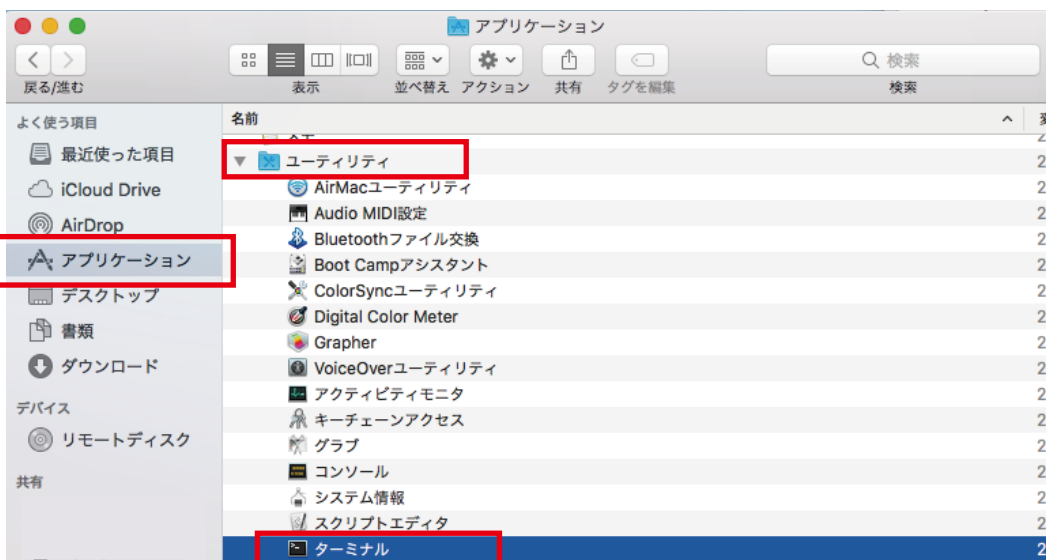


- ③ コマンドプロンプトに「ipconfig」と入力すると、IPアドレスを確認できます。



Mac 環境

- ① Finder を起動して「アプリケーション」、「ユーティリティ」を選択し、「ターミナル」をダブルクリックします。



- ② ターミナルで「networksetup -listallhardwareports」と入力すると、ネットワークハードウェアの一覧が表示されます。

- ③ 一覧から IP アドレスを調べたいハードウェアのインターフェース名（「Device」の項目）を確認します。

```
% networksetup -listallhardwareports
Hardware Port: Wi-Fi
Device: en0
Ethernet Address:
```

- ④ 調べたいハードウェアのインターフェース名が「en0」の場合、「ifconfig en0」と入力すると、ネットワーク情報が出力されます。「inet」の値が IP アドレスです。

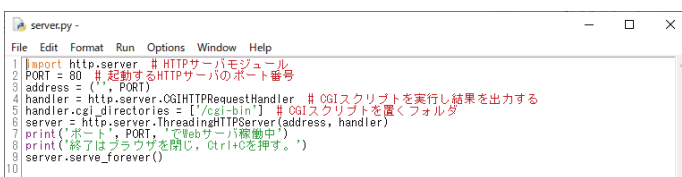
```
% ifconfig en0
en0: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST>
options=6463<RXCSUM, TXCSUM, TS04, TS06, CHANNEL_...>
ether
inet6 fe80::85d:c069:3b98:b7b0%en0 prefixlen 64
inet 192.168.0.18 netmask 0xfffff00 broadcast 192.168.0.255
nd6 options=201<PERFORMNUD,DAD>
media: autoselect
status: active
```

2. サーバを起動するプログラムの作成

- ① サーバの役割をする PC で IDLE を使用して下記の server.py を作成します。

server.py

```
import http.server # HTTP サーバモジュール
PORT = 80 # 起動する HTTP サーバのポート番号
address = ('', PORT)
handler = http.server.CGIHTTPRequestHandler # CGI スクリプトを実行し結果を出力する
handler.cgi_directories = ['/cgi-bin'] # CGI スクリプトを置くフォルダ
server = http.server.ThreadingHTTPServer(address, handler)
print('ポート', PORT, 'で Web サーバ稼働中')
print('終了はブラウザを閉じ, Ctrl+C を押す。')
server.serve_forever()
```



IDLE で作成したときの画面

- ② 「root」という名前のフォルダを作成し、①で作成した「server.py」ファイルを格納します。

(※フォルダ名は任意に設定可能)



3. 自動実行ファイルの作成

本テキストでは Web サーバの起動方法（server.py の実行方法）として自動実行ファイルを使用します。自動実行ファイルは自動的に実行するコマンド（命令）を記述したファイルです。ダブルクリックするだけで、そのファイルに記述されたコマンド（命令）を実行でき、簡単に Web サーバを起動できます。

Windows 環境

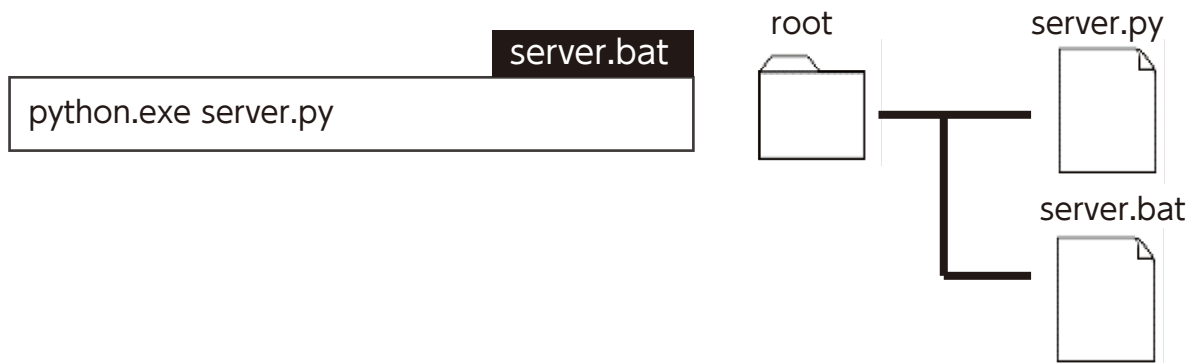
Python をインストールする際の「Add Python xx to PATH」の横のチェック（☒）の有無で手順が異なります。（準備①-1-②参照）

●チェック（☒）を入れた場合（パス設定をした場合）

下記の server.bat を作成し、server.py と同じ階層に格納します。

server.bat はメモ帳などのテキストエディタで作成し、拡張子を「bat」に変更してファイル名を server.bat にして保存します。

メモ帳で server.bat を開く場合は、ファイルの種類を「すべてのファイル (*.*)」に変更してから開きます。



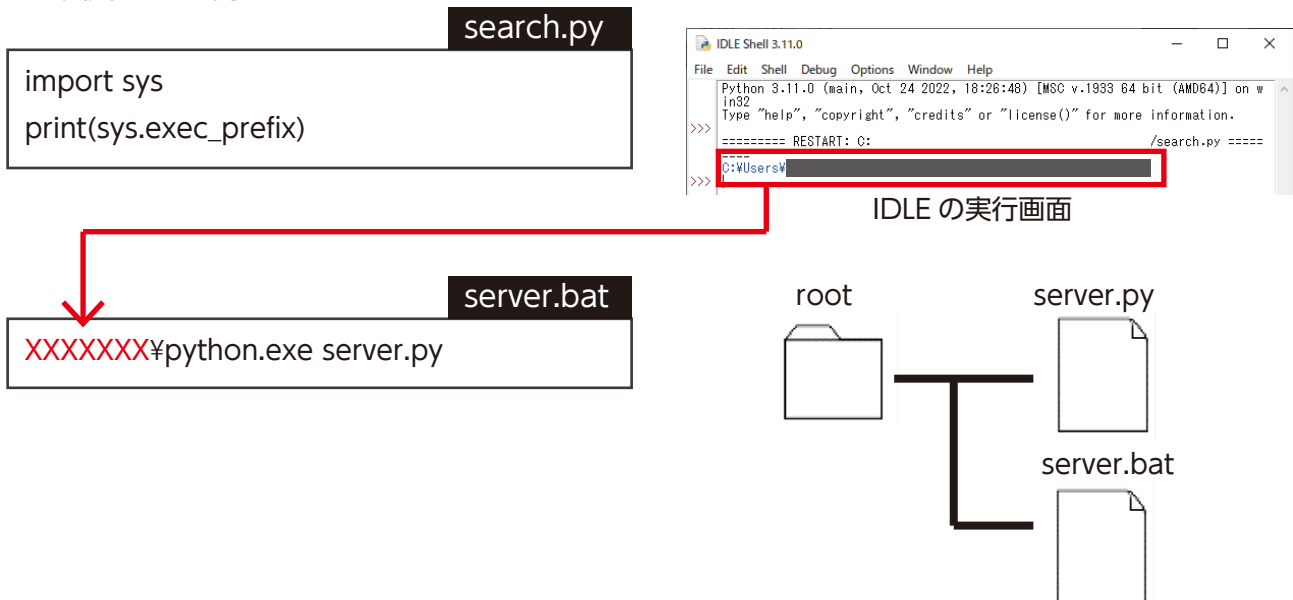
●チェック (☑) を入れなかった場合 (パス設定をしなかった場合)

Python の実行ファイル (Python.exe) の場所を調べるために下記の search.py を作成し、IDLE で実行します。

表示されたパス名 (赤枠) を下記の server.bat の XXXXXXXX に入力して、server.bat を作成します。

作成した server.bat は server.py と同じ階層に格納します。

server.bat はメモ帳などのテキストエディタで作成し、拡張子を「bat」に変更してファイル名を server.bat にして保存します。メモ帳で server.bat を開く場合は、ファイルの種類を「すべてのファイル (*.*)」に変更してから開きます。

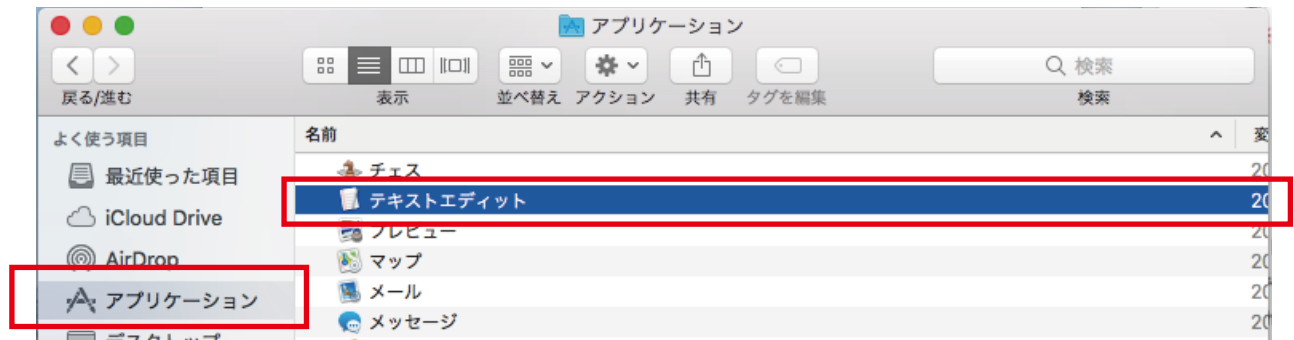


なお、P.4 でダウンロードしたプログラム内の server.bat はパス設定をした場合のファイルになっています。パス設定をしなかった場合は server.bat を環境に合わせて修正してください。

Mac 環境

本テキストでは server.py の実行方法として、自動実行ファイル server.command を使用します。

- ①server.command はテキストエディットを使用して作成します。Finder を起動して「アプリケーション」、「テキストエディット」をクリックしてください。



- ②テキストエディットを起動後、メニューの「フォーマット」をクリックして「標準テキストにする」を選択してください。

- ③下記の server.command を作成し、server.py と同じ階層に格納します。

server.command

```
#!/bin/sh
cd `dirname $0`
python3 server.py
```

root

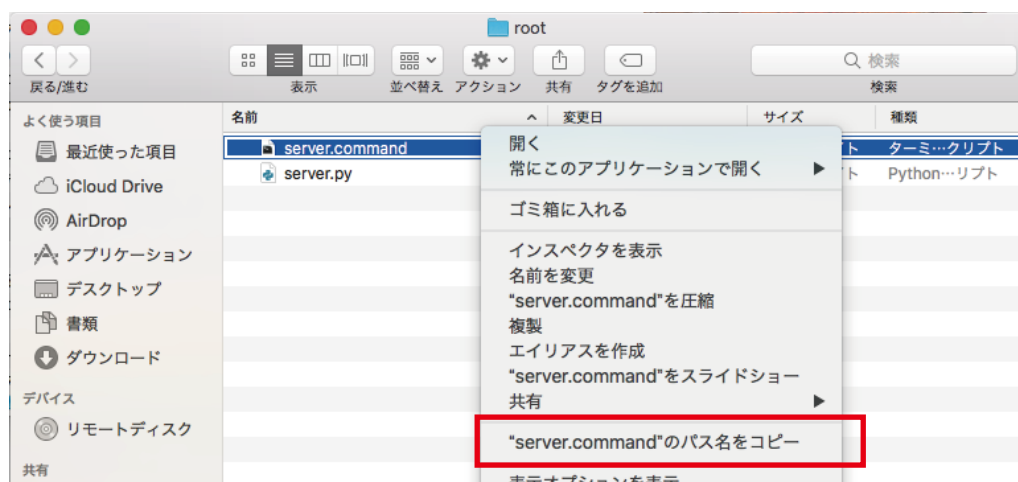
server.py

server.command

※「dirname \$0」の前後にある「`」はバッククォーテーションです。キーボードの Shift キーを押しながら @ キーを押して入力してください。

- ④command ファイルを実行するには、アクセス権限を変更する必要があります。server.command のパーミッション（権限）を 755 に変更します。

Finder で server.command が置かれているフォルダに移動し、server.command を右クリックします。メニューが表示されたら、option キーを押して「server.command」のパス名をコピー」を選択してください。



⑤演習①「1.IP アドレスの確認」を参考にターミナルを起動して、「chmod 755 ㊤」と入力し、Enter キーを押してください。㊤の部分には④でコピーした server.command のパス名を貼りつけます。

※server.command がユーザーのホームディレクトリにおかれている場合は、/Users/ ユーザー名 /server.command が張り付けられるので下の例のようになります。

例：chmod 755 /Users/ ユーザー名 /server.command

4. テストプログラムの作成

① Web サーバの動作を確認する下記のプログラムを IDLE で作成します。

test.py

```
import sys, io
html = '''
<!DOCTYPE html>
<html lang="ja">
  <head><meta charset="UTF-8"></head>
  <body>
    Web サーバ稼働中
  </body>
</html>
'''

sys.stdout = io.TextIOWrapper(sys.stdout.buffer, encoding='utf-8')
print('Content-Type: text/html; charset = UTF-8¥n¥n')
print(html)
```

Mac 環境のみ先頭行に以下のコードを追加してください。

```
#!/usr/bin/env python3
```

●シェバンについて

Windows 環境では基本的に Python の実行パスの記述は不要ですが、Mac 環境で実行する場合は、実行パスの記述が必要なため、py ファイルの先頭行に上記のコード「#!/usr/bin/env python3」を追加します。

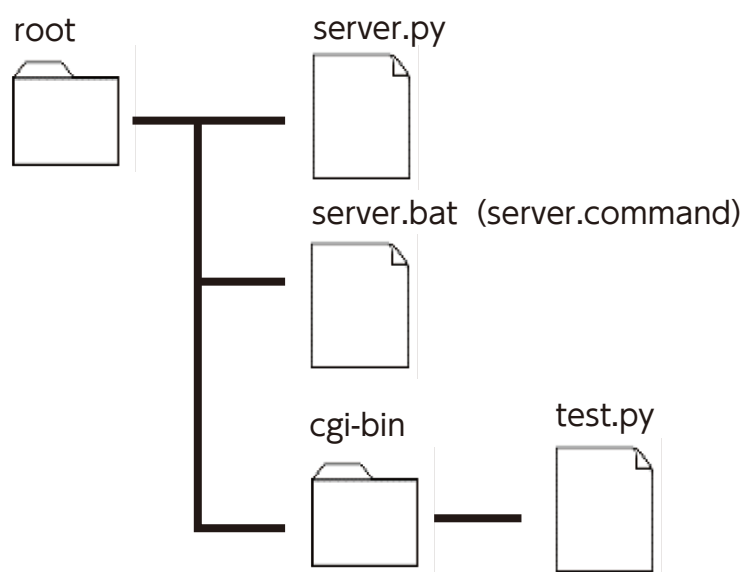
これをシェバンといいます。

シェバンはプログラムを実行する処理系（インタプリタ）を指定する仕組みです。

ただし、IDLE や VSCode 等の Python の実行環境で py ファイルを実行させる場合では、実行環境の中で処理系を指定しているため、シェバンの記述は不要です。

②「cgi-bin」という名前のフォルダを root フォルダ下に作成し、①で作成した「test.py」ファイルを cgi-bin フォルダ下に格納します。

(※フォルダ名は任意に設定可能)



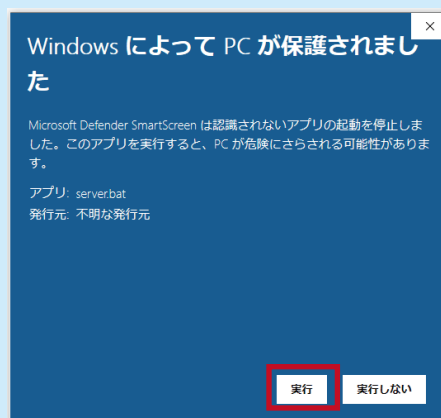
5. Web サーバの動作確認

① sever.bat (Mac の場合は server.command) をダブルクリックすると、server.py が自動的に実行され、下記のような画面が表示されます。

```
C:\WINDOWS\system32\cmd.exe
C:\Users\>python.exe server.py
ポート 80 でWebサーバ稼働中
終了はブラウザを閉じ、Ctrl+Cを押す。
```

Windows 環境

bat ファイルをダブルクリックしたときに Microsoft Defender SmartScreen が開き、「Windows によって PC が保護されました」と表示される場合は「詳細情報」→「実行」の順にクリックすると、上記ウィンドウが立ち上がります



●Webサーバが起動しない場合のトラブルシューティング

server.pyのプログラムではポート番号の指定を80としています。Webサーバが起動しない場合はポート番号80が他で使用されている可能性があります。このような場合はポート番号80を使用している他のアプリケーションを停止するか、ポート番号を8080など使用されていないポート番号に変更してください。

ただし、80以外のポート番号を指定した場合(ここでは8080)は、次項②でのアドレス入力時に、「http://localhost:**8080**/cgi-bin/test1.py」や、次項③でのアドレス入力時に「http://IPアドレス:**8080**/cgi-bin/test1.py」というように、常にポート番号を指定する必要があります。

また、server.py (p.21) の2行目の「PORT=80」の部分を「PORT=**8080**」に変更する必要があります。

Windows 環境

[ポートの使用状況の確認方法]

演習④「1.IPアドレスの確認」を参考にコマンドプロンプトを起動させ、以下のように入力します。

netstat -nao | findstr 80

```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.18363.657]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\GALLERYA01>netstat -nao | findstr 80
TCP 0.0.0.0:80 0.0.0.0:0 LISTENING 4
TCP 0.0.0.0:1801 0.0.0.0:0 LISTENING 4264
TCP 0.0.0.0:5040 0.0.0.0:0 LISTENING 8024
TCP 0.0.0.0:8080 0.0.0.0:0 LISTENING 4
```

左の列からそれぞれ、プロトコル、ローカルアドレス、外部アドレス、状態、PIDを表しています。上記画像のように左から2列目のローカルアドレスのコロン[:]の後が80になっている場合では80のポート番号は、すでに使われている状態(LISTENING)を表しており、このままでは使用できない状態だということがわかります。上の例では、80のポート番号を使用しているプログラムのプロセスID (PID) が4であることがわかります。どのプログラムが80のポート番号を使用しているかを調べるにはCtrl+shift+Escを押して、タスクマネージャーを起動させ、「詳細」タブで確認することができます。ここではPIDが4のプロセス(実行中のプログラム)がSystemであることがわかります。

プロセス	パフォーマンス	アプリの履歴	スタートアップ	ユーザー	詳細	サービス
名前	PID	状態	ユーザー名	CPU	メモリ(アクテ...	UACの仮想
System Idle Process	0	実行中	SYSTEM	99	8 K	
System	4	実行中	SYSTEM	00	20 K	
svchost.exe	8	実行中	NETWORK...	00	8,692 K	不許可
Regstrv	120	実行中	SYSTEM	00	13,140 K	不許可

ポート番号80を使用しているサービスの一つとして、マイクロソフト社が提供しているIIS (Internet Information Services/インターネット インフォメーション サービス)というものがあります。前記のように「System」によってポート番号80を使用されている場合は、IISによるものの可能性があります。下記のようなアプリケーションソフトをインストールされているとIISが同時にインストールされ実行されている場合があります。

WebMatrix

Visual Studio

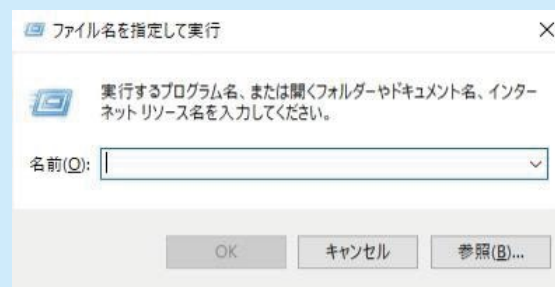
Windows Remote Management

World Wide Web Publishing Service

「System」を停止するためには、以下の方法で「World Wide Web Publishing Service」を停止してください。

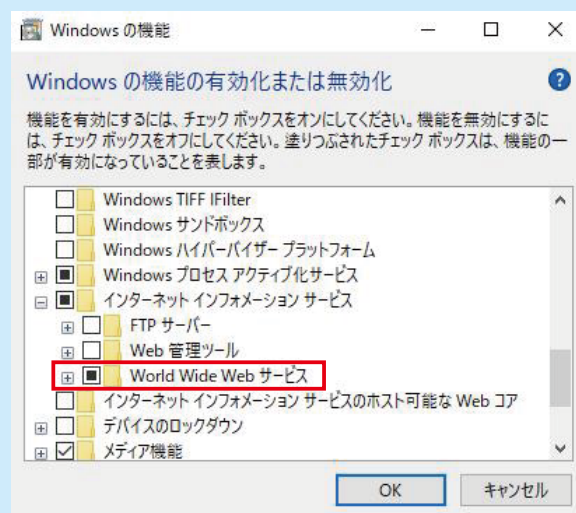
①コントロールパネルを開いてください。

「Windowsキー」+「R」で以下のウインドウを開き、「control」と入力



②「表示方法」を「カテゴリ」に変更し、「プログラム」→「プログラムと機能」

→「Windows の機能の有効化または無効化」を選択し、「Windows の機能」ダイアログを表示させ、「インターネット インフォメーション サービス」の項目を開いて、「World Wide Web サービス」のチェックを外してください。



Mac 環境

<ポートの使用状況の確認>

演習④1.IP アドレスの確認を参考にターミナルを起動して以下のように入力します。

lsof -P -i:80

```
~ % lsof -P -i:80
COMMAND PID USER  FD  TYPE    DEVICE  SIZE/OFF  NODE NAME
Python  370  user   4u  IPv4  0xf3a63fd      0t0  TCP *:80 (LISTEN)
```

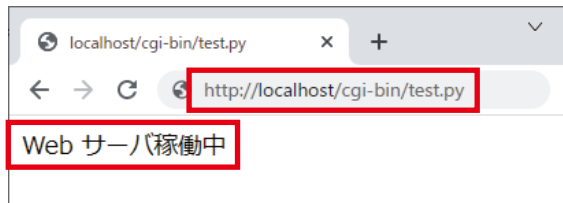
最後の列 (NAME) のローカルアドレスのコロン「:」の後が「80」になっているものがある場合は、ポート番号 80 を使用しているアプリケーションがすでに存在していることを表しています。ポート番号を 8080 など使用されていないポート番号に変更してください。

②サーバの役割をする PC で Web ブラウザを開き、アドレス欄に「http://localhost/cgi-bin/test.py」と入力することで、「Web サーバ稼働中」と表示されることを確認してください。

80 以外のポート番号を指定した場合は

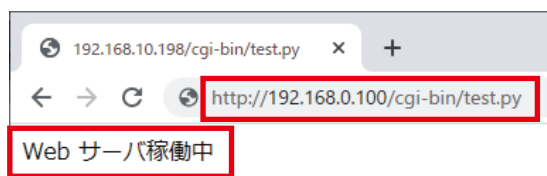
「http://localhost: **ポート番号** /cgi-bin/test.py」

と入力してください。



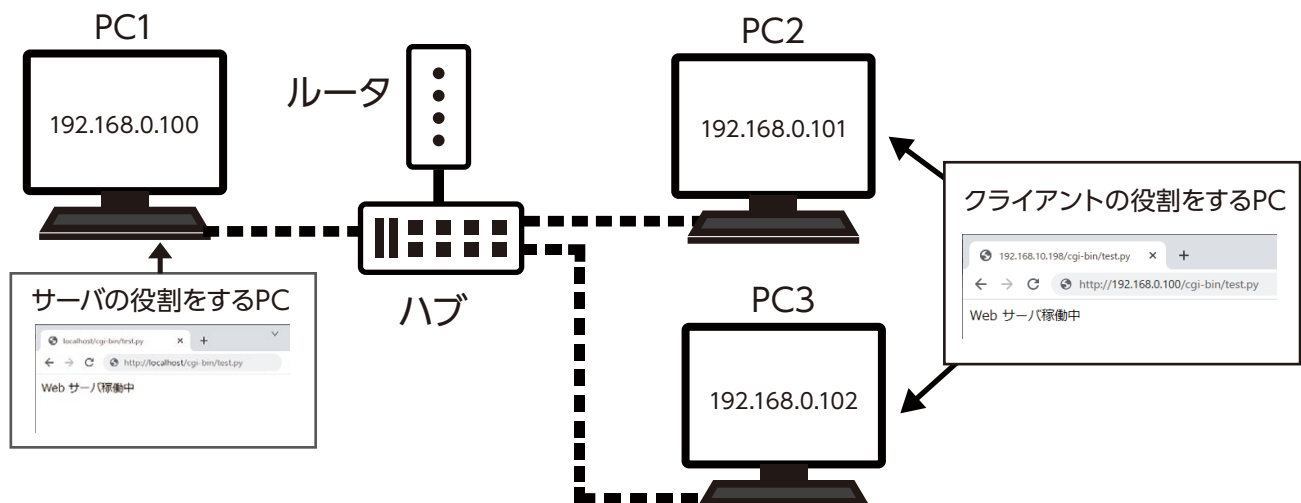
③LAN 接続された他の PC で Web ブラウザを開き、アドレス欄に「http://**IP アドレス** /cgi-bin/test.py」と入力し、同様に「Web サーバ稼働中」と表示されるか確認してください。**(※IP アドレス=test.pyが入っている Web サーバを起動した PC の IP アドレスを入力)**

80 以外のポート番号を指定した場合は「http://IP アドレス : **ポート番号** /cgi-bin/test.py」と入力してください。



ネットワーク環境例 ※IP アドレスの数値は一例

図中のルータやハブ、PC1、PC2、PC3 は、コンピュータ教室に設置されている既存の装置を使用します。



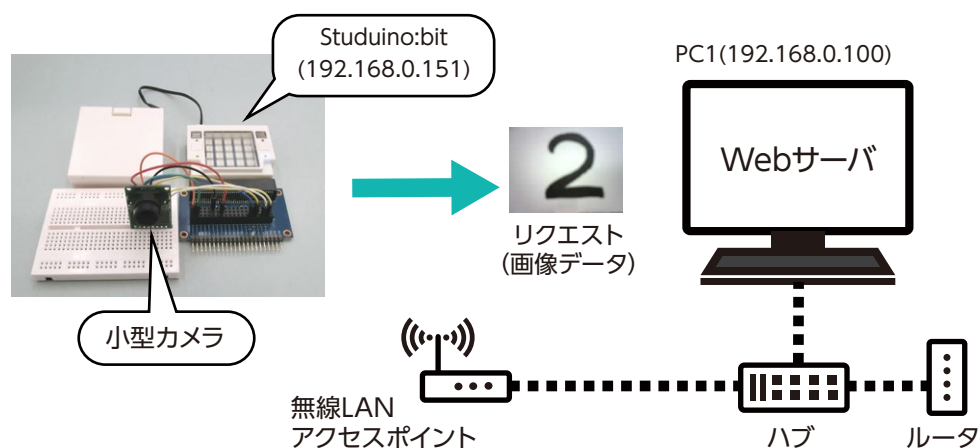
④Web サーバを終了する際は、①の画面で「Ctrl キー+C キー」を押します。

Windows 環境では続いて、「y」を入力して「Enter キー」を押してください。

演習② 小型カメラの制御

小型カメラをStuduino:bitに接続して、撮影し、撮影された画像データをWebサーバへ送信します。

※IP アドレスの数値は一例



- 無線LAN アクセスポイントとしてWi-Fiルータを使用する場合は、Wi-Fiルータの取扱説明書を参照のうえ、ルータモード(RTモード)をOFFにして、アクセスポイントモード(APモード)に設定してください。
- 無線LAN アクセスポイントとしてモバイルホットスポットを利用する場合は、付録Eを参照してください。
- モバイルホットスポットとは、有線LAN で接続されたWindows10 やWindows11 のコンピュータをWi-Fiルータ化する機能です。なお、図中のルータやハブ、PC1は、コンピュータ教室に設置されている既存の装置を使用します。ハブの空きポートに無線LANアクセスポイントをLANケーブルで接続し、これにStuduino:bitを無線で接続します。

1. 小型カメラを使用するために必要なライブラリを Studuino:bit に転送する。

①下記より zip ファイルをダウンロードして、解凍してください。

https://www.artec-kk.co.jp/dl/93275/data/camera_program.zip

解凍すると下記の6点のファイルが入っています。

ファイル名	役 割
capture.py	カメラで撮影する
transfer.py	カメラで撮影した画像をサーバへPOST 送信し、応答を得る
urequests_l.py	サーバにリクエストを投げて、応答を得るためのモジュール
ov2640.py	カメラを制御するモジュール
ov2640_constants.py	カメラを制御するレジスタ情報
ov2640_lores_constants.py	カメラの画角等を設定するレジスタ情報

②ダウンロードしたファイルを mu_code フォルダに格納します。

Windows の場合 C:\Users\ ユーザ名 \mu_code

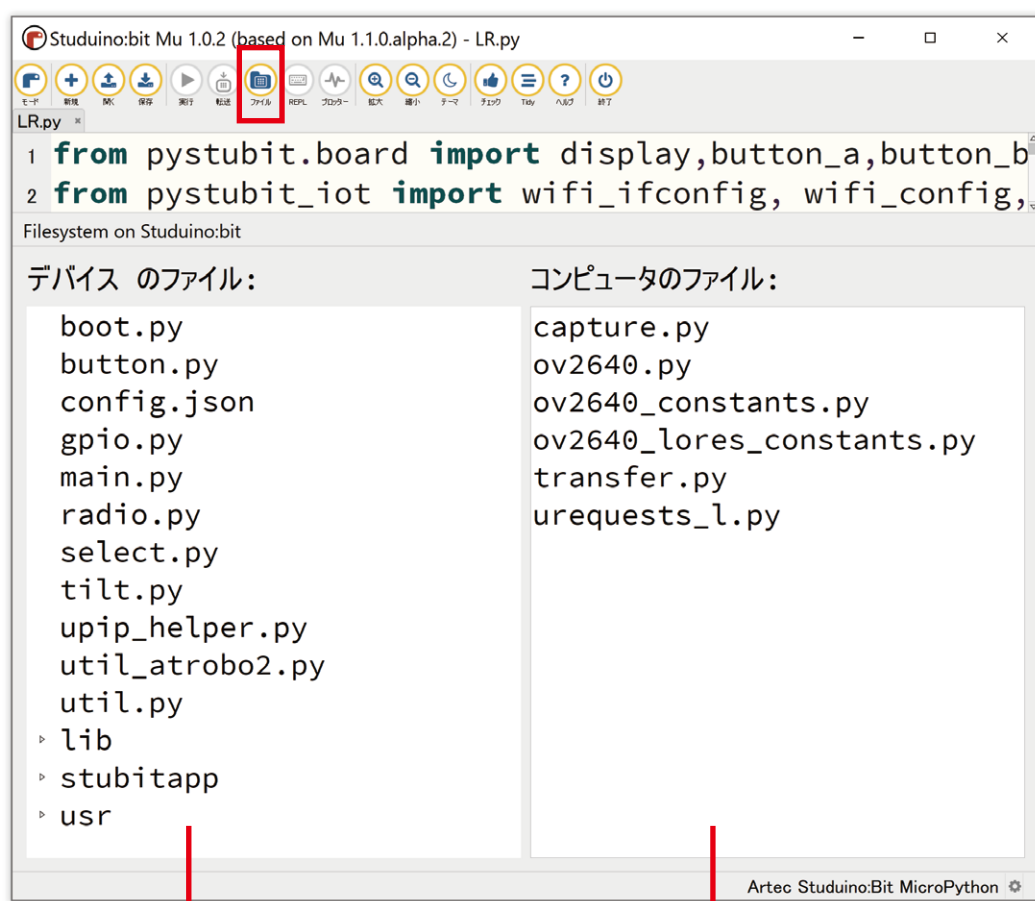
Mac の場合 /Users/ ユーザ名 /mu_code

③Studuino:bit と PC を接続します。

④Muエディタを使って、Studuino:bitにダウンロードしたファイルを転送します。

「ファイル」ボタンをクリックすると下側にウィンドウが表示されます。

※ウィンドウ表示中は「実行」ボタン、「転送」ボタン、「REPL」ボタン、「プロッター」ボタンは灰色表示になり使用できません。再度「ファイル」ボタンをクリックすることでウィンドウが非表示になり、他のボタンを使用できるようになります。

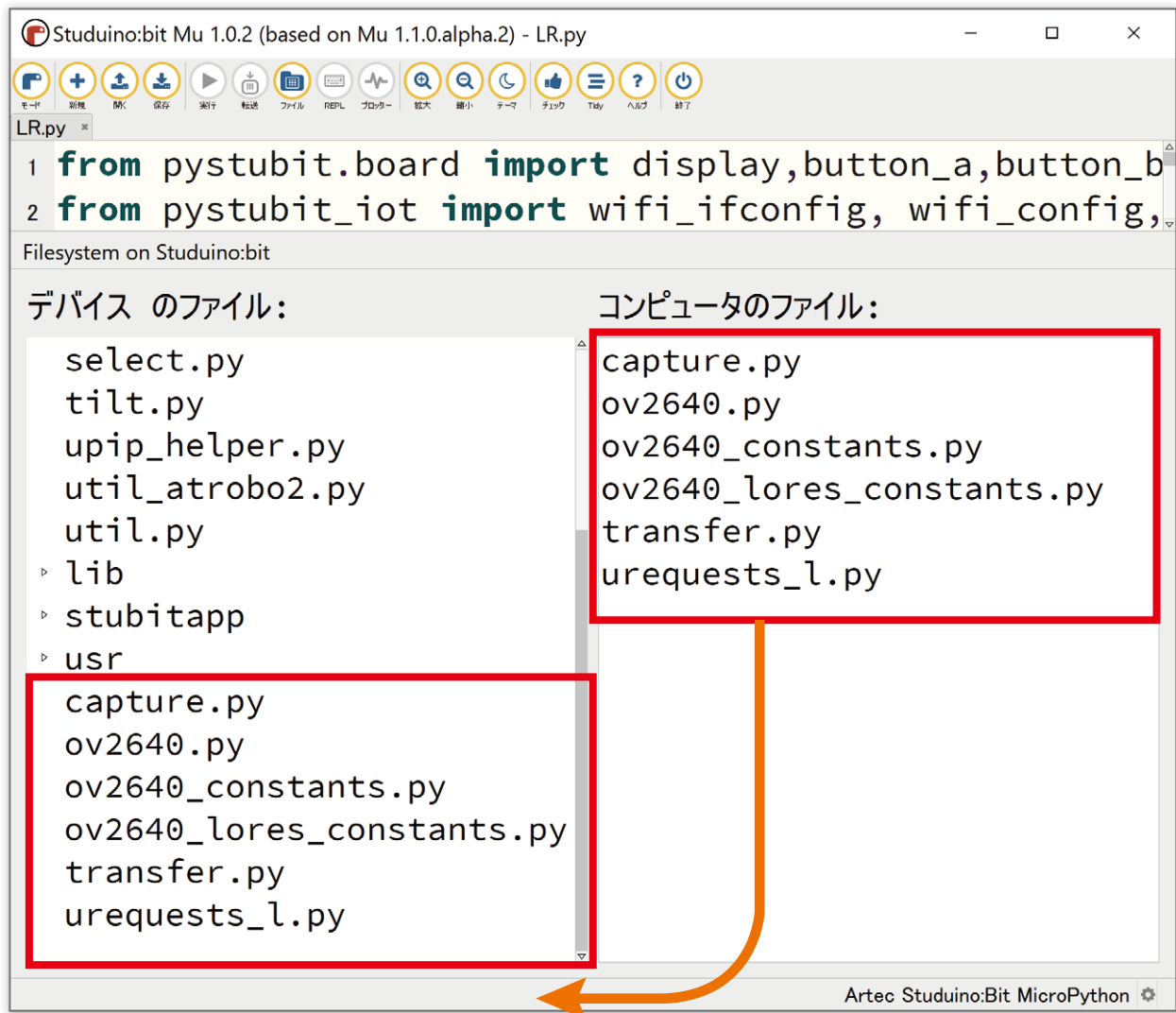


Studuino:bit の中のファイル

PC の mu_code フォルダの中のファイル

⑤ダウンロードしたファイルを 1 つずつ、デバイスのファイル直下にドラッグ&ドロップして Studuino:bit にファイルを転送（コピー）します。

※「lib」や「stubitapp」、「usr」に向けてドラッグ&ドロップするとそれらのフォルダ内にファイルが転送されてしまいます。デバイスのファイル直下である「boot.py」などに向けてドラッグ&ドロップしてください。



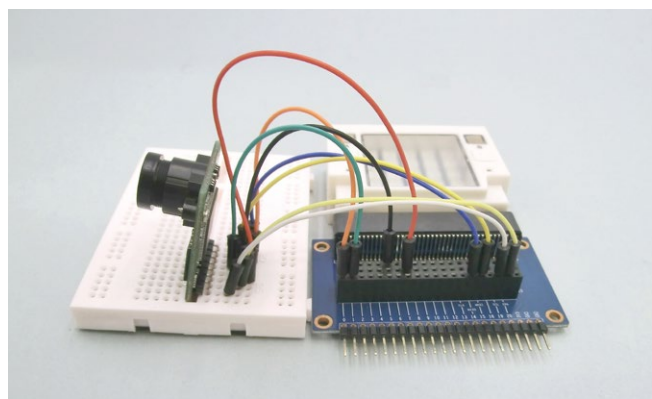
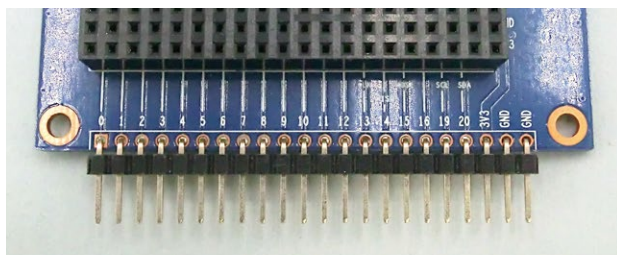
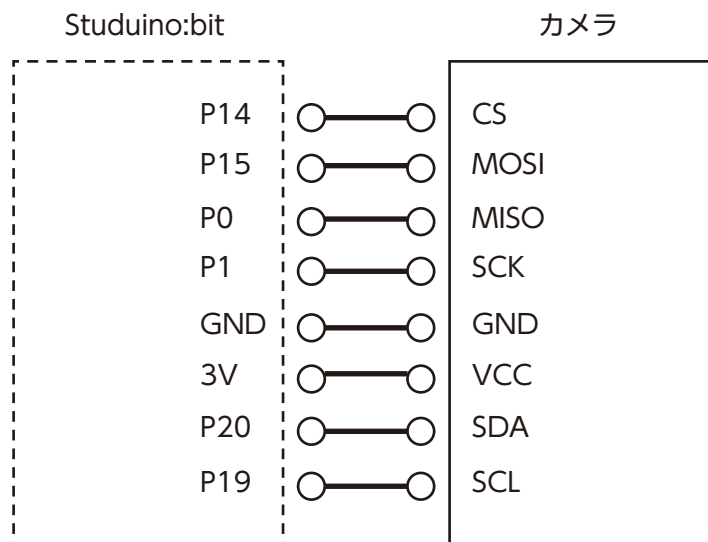
ドラッグ&ドロップ

2.Studuino:bit 側のプログラム作成

①Studuino:bitとカメラを接続します。

ブレッドボードの使い方詳細は付録Aを参照してください。

変換エッジコネクタの使い方詳細は付録Bを参照してください。



②Studuino:bitとPCを接続します。

③下記のcap.pyをMuエディタで作成し、Studuino:bitに転送します。

Wi-Fiが未接続状態のときBボタンが押されるとWi-Fiに接続し、接続状態のときに押されると接続を解除します。

Aボタンで画像を撮影後、サーバへ撮影した画像送信します。

displayオブジェクトの機能詳細は付録Cを参照してください。

Wi-Fiオブジェクトの機能詳細は付録Dを参照してください。

cap.py

```
from pystubit.board import display,button_a,button_b
from pystubit_iot import wifi_ifconfig, wifi_config, wifi_connect, wifi_disconnect
import capture
import transfer
import time
# 画像の送信先の指定
url = 'http://Web サーバの IP アドレス : ポート番号 /cgi-bin/device2_1.py'
fname = 'image.jpg'
wifi_isconnected = 0
wifi_config(ssid='アクセスポイントの SSID', pwd='パスワード')
while True:
    if button_b.is_pressed():
        if wifi_isconnected == 0:
            # Wi-Fi に接続する
            res = wifi_connect(trytime=5)
            wifi_ifconfig(('A', 'B', 'C', 'D')) #Studuino:bit に固定の IP アドレスを割り振る
            wifi_isconnected = 1
            time.sleep(1)
        elif wifi_isconnected == 1:
            # Wi-Fi 接続を解除する
            wifi_disconnect()
            wifi_isconnected = 0
            time.sleep(1)
    if button_a.is_pressed():
        time.sleep(1)
        display.clear()
        display.set_pixel(2,2,(10,0,0))
```

※次のページに続く

※前のページの続き

```
# カメラで撮影し、画像ファイル fname を作成する
```

```
capture.run(fname)
```

```
display.set_pixel(0,2,(0,10,0,))
```

```
display.set_pixel(2,2,(0,10,0,))
```

```
display.set_pixel(4,2,(0,10,0,))
```

```
ans = 'false'
```

```
# 画像ファイル fname をサーバへ送信する
```

```
while ans == 'false':
```

```
    ans = transfer.run(url, fname)
```

```
display.clear()
```

※wifi_ifconfig 関数内の各記号には右記の
情報を入力してください。(IP アドレス
を自動で割り当てることができる DHCP
の場合は、この関数は不要です。)

記号	アドレス
Ⓐ	Studuino:bit に割り振る IP アドレス
Ⓑ	サブネットマスク
Ⓒ	デフォルトゲートウェイ
Ⓓ	DNS サーバ

●wifi_config関数について

無線LANアクセスポイントとしてWi-Fiルータを使用する場合は、wifi_config関数の引数にWi-Fiルータの本体もしくは取扱説明書に記載されているSSIDとパスワードをそれぞれ設定してください。無線LANアクセスポイントとしてモバイルホットスポットを使用する場合は、付録Eを参照してください。

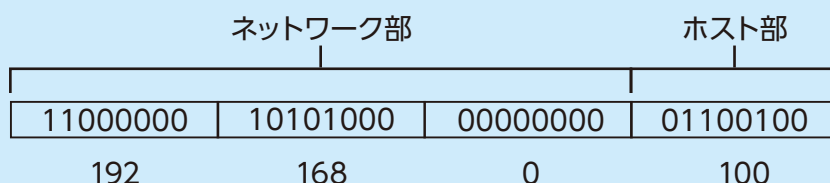
```
wifi_config(ssid=' アクセスポイントの SSID', pwd=' パスワード ')
```

●wifi_ifconfig関数について

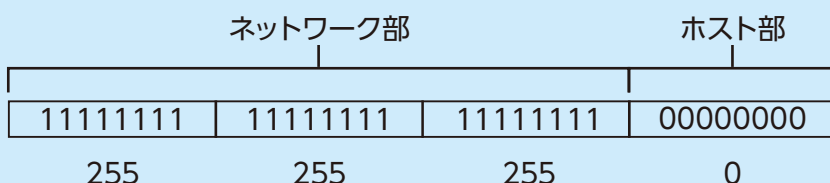
- (1) IPアドレスのネットワーク部とホスト部IPアドレスは、上位のネットワーク部と下位のホスト部から成り立っています。ネットワーク部は、そのIPアドレスが属しているセグメント(ネットワークの最小のまとまり)を識別するための部分で、ホスト部は、ネットワーク内のコンピュータを識別するための部分です。同一セグメント内ではネットワーク部はすべて等しくなります。例えば、256個のIPアドレスが利用できるクラスCでは、上位の24ビットがネットワークアドレス、下位の8ビットがホストアドレスになります。

<例>

ネットワーク部が24ビットのIPアドレスが「192.168.0.100」の場合、ネットワーク部は「192.168.0」、ホスト部は「100」になります。



ネットワーク部とホスト部の区別は、サブネットマスクを調べることによってわかります。サブネットマスクとは、ネットワーク部とホスト部を区別するための数値のことで、サブネットマスクの1の部分はIPアドレスのネットワーク部を、サブネットマスクの0の部分はIPアドレスのホスト部をそれぞれ表しています。サブネットマスクが255.255.255.0の場合は、先頭24ビットがネットワーク部で、残り8ビットがホスト部になります。



(2) wifi_ifconfig関数とその引数の確認・設定方法

wifi_ifconfigは、Studuino:bitに固定IPアドレスを設定する関数ですが、引数はタプルで指定します。

wifi_ifconfig(('A','B','C','D'))

記号	アドレス
①	Studuino:bit に割り振る IP アドレス
②	サブネットマスク
③	デフォルトゲートウェイ
④	DNS サーバ

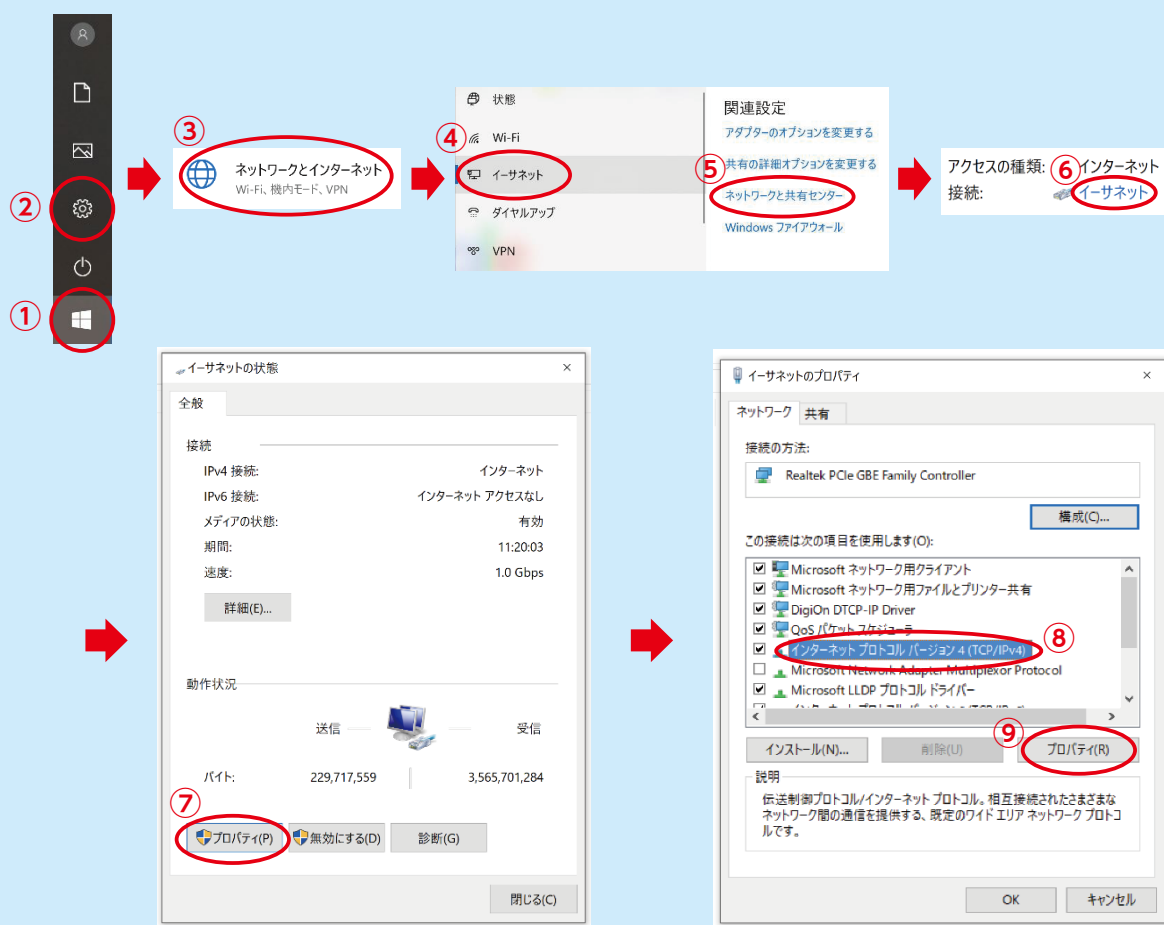
①、②、③、④の各アドレスは以下のように設定します。

[コンピュータ教室の各アドレスの確認方法]

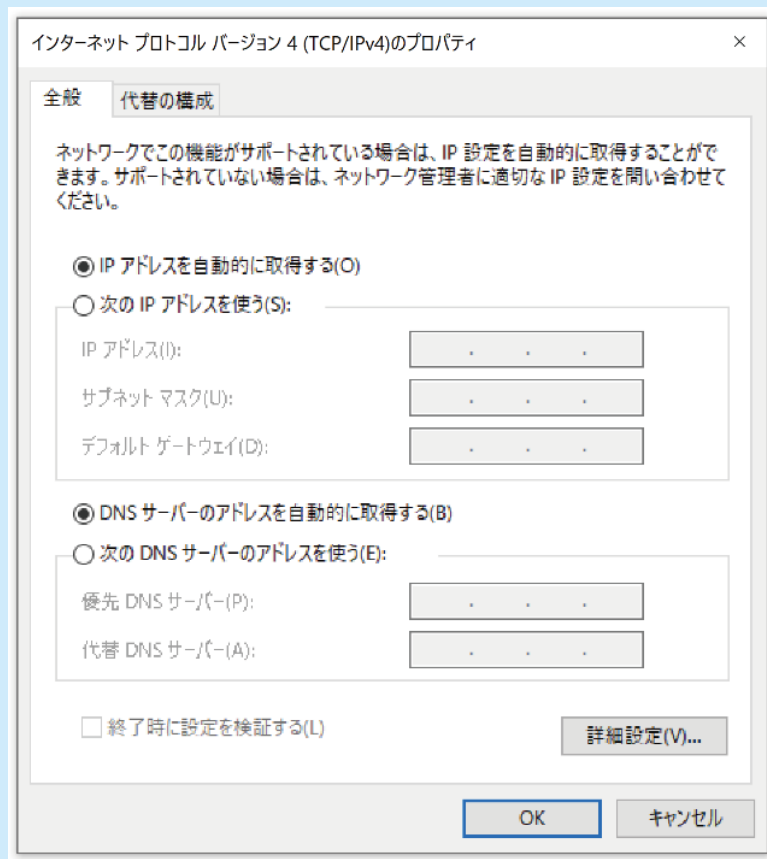
Windows 環境

コンピュータ教室のLAN内にある任意のPCを起動します。

①スタートボタン → ②設定 → ③ネットワークとインターネット → ④イーサネット → ⑤ネットワークと共有センター → ⑥イーサネット → ⑦プロパティ → ⑧インターネットプロトコルバージョン4(TCP/IPv4) → ⑨プロパティの順にクリックします。



※インターネットプロトコルバージョン4(TCP/IPv4)のプロパティの画面が以下のように表示された場合は、LAN内の装置にはIPアドレスが自動で割り当てられています(自動で割り当てる仕組みをDHCPと呼びます)。このような場合は、プログラムの中のwifi_ifconfig関数がある部分を削除してください。

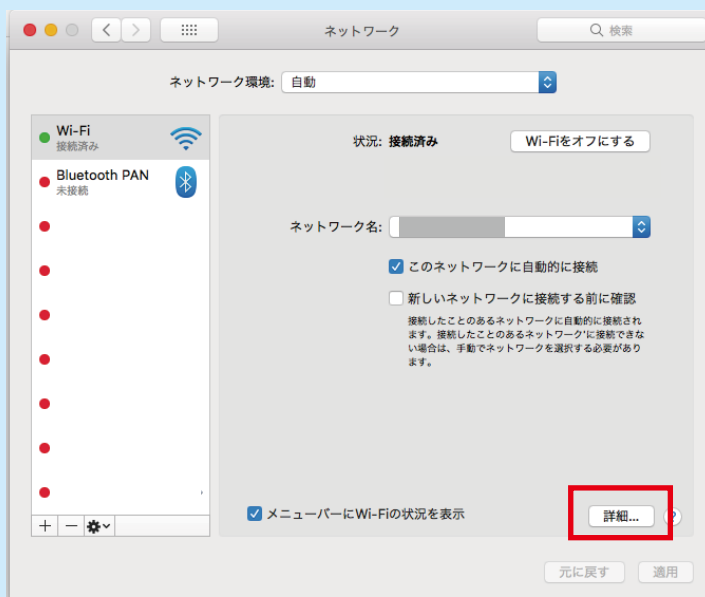


Mac 環境

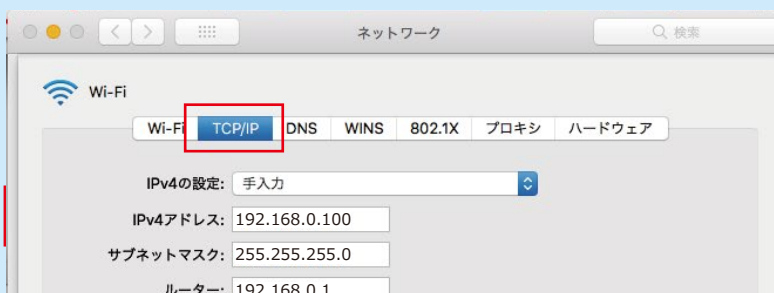
「Appleメニュー」(画面左上リンゴのマーク)-「システム環境設定」-「ネットワーク」をクリックします。



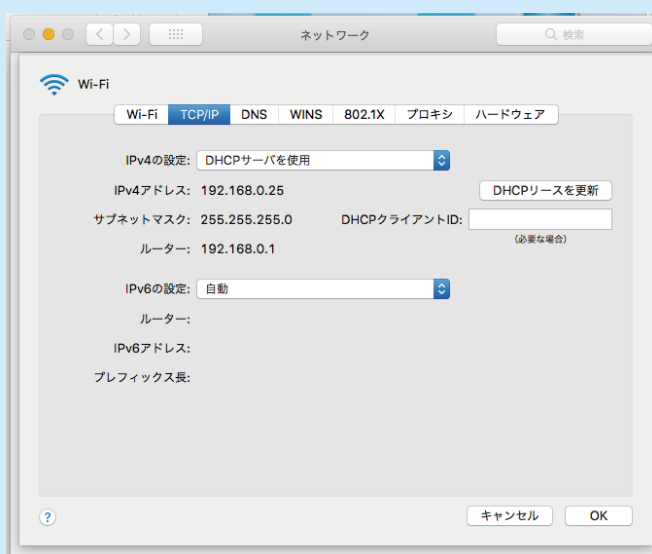
左側の項目から、コンピュータ教室のLANに接続している「Wi-Fi」(無線LAN)または、「Ethernet」(有線LAN)を選択し、「詳細」ボタンをクリックします



「TCP/IP」タブをクリックします。



※以下のような画面が表示された場合は、LAN 内の装置には IP アドレスが自動で割り当てられています (自動で割り当てる仕組みを DHCP と呼びます)。このような場合は、プログラムの中の wifi_ifconfig 関数がある部分を削除してください。



Windows 環境

※⑫と⑬が同じ値になっていることがあります。
これはルータの DNS サーバの機能を使用しているためです。

Mac 環境

⑩のアドレスの指定例： 192.168.0.***

同一セグメント内で割り当てられている
IP アドレスのネットワーク部を指定

コンピュータ教室内の他の装置に割り当て
られているホスト部と重ならない値を指定

「IP アドレス」の⑩の部分に表示されるアドレス（この例では 192.168.0.100）は、あくまで現在使用しているパソコンの IP アドレスです（※この例ではサブネットマスクが 255.255.255.0 であることから、IP アドレスのネットワーク部は 192.168.0 で、ホスト部は 100 です）。

p.38の(2)の④のStuduino:bit に割り振るIPアドレスは、同一セグメント内で重ならない値を割り当てる必要があります。

④のIPアドレスのネットワーク部には、コンピュータ教室内の各装置に割り当てられているIPアドレスのネットワーク部と同じ値を使用します。

④のIPアドレスのホスト部には、コンピュータ教室内の他の装置に割り当てられているホスト部と重ならない値を使用します。

事前に同じLAN（コンピュータ教室）内にある他のパソコンやプリンタ、イメージスキャナーなどの装置に割り当てられているIPアドレスを調査してください（LAN を構築した時の設定書などを調査します）。

3. サーバ側のプログラム作成

①下記のdevice2_1.pyをIDLEで作成します。

device2_1.pyはStuduino:bitから画像データを受信して、temp.jpgに保存します。

※Python3.13以降のバージョンで実行する場合はcgiモジュールが使用できないため、付録Hを参照してください。

Windows 環境

device2_1.py

```
import cgi
# ファイルにバイナリモードでデータを書き込む関数
def fwriteb(data,fn):
    file = open(fn, mode='wb')
    file.write(data)
    file.close()
print('Content-Type: text/plain¥n')
ftemp= 'cgi-bin/temp.jpg' # Studuino:bit から受信したデータ
param = cgi.FieldStorage() # Studuino:bit から送られたパラメータを取得
val = param.getvalue('img') # キー img から値 val (画像データ) を取り出す
fwriteb(val, ftemp) # img の値をファイルに書き込む
```

Mac 環境

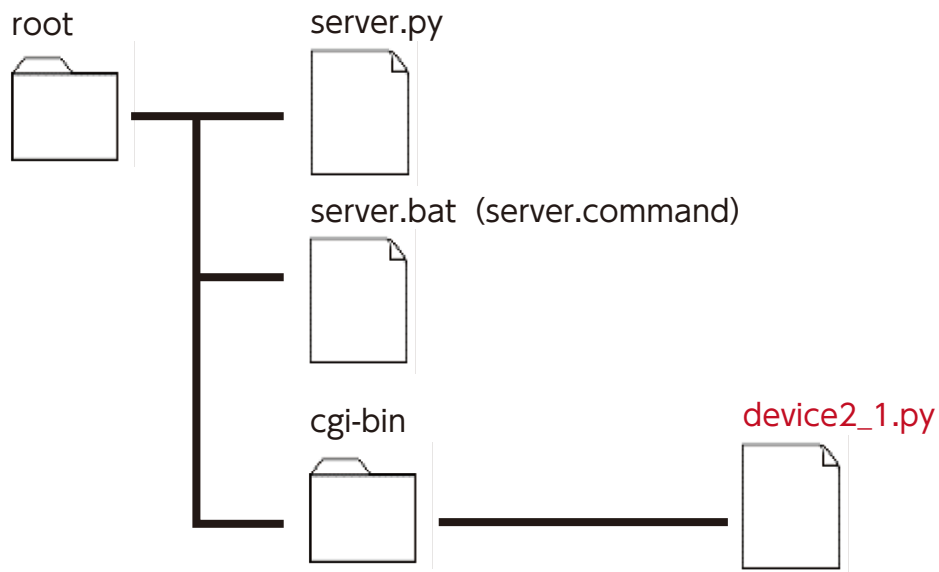
device2_1.py

```
#!/usr/bin/env python3
import cgi
# ファイルにバイナリモードでデータを書き込む関数
def fwriteb(data,fn):
    file = open(fn, mode='wb')
    file.write(data)
    file.close()
print('Content-Type: text/plain¥n', end='¥r¥n')
ftemp= 'cgi-bin/temp.jpg' # Studuino:bit から受信したデータ
param = cgi.FieldStorage() # Studuino:bit から送られたパラメータを取得
val = param.getvalue('img') # キー img から値 val (画像データ) を取り出す
fwriteb(val, ftemp) # img の値をファイルに書き込む
```

※「¥」は環境によってバックスラッシュ「\」が表示されることがあります。

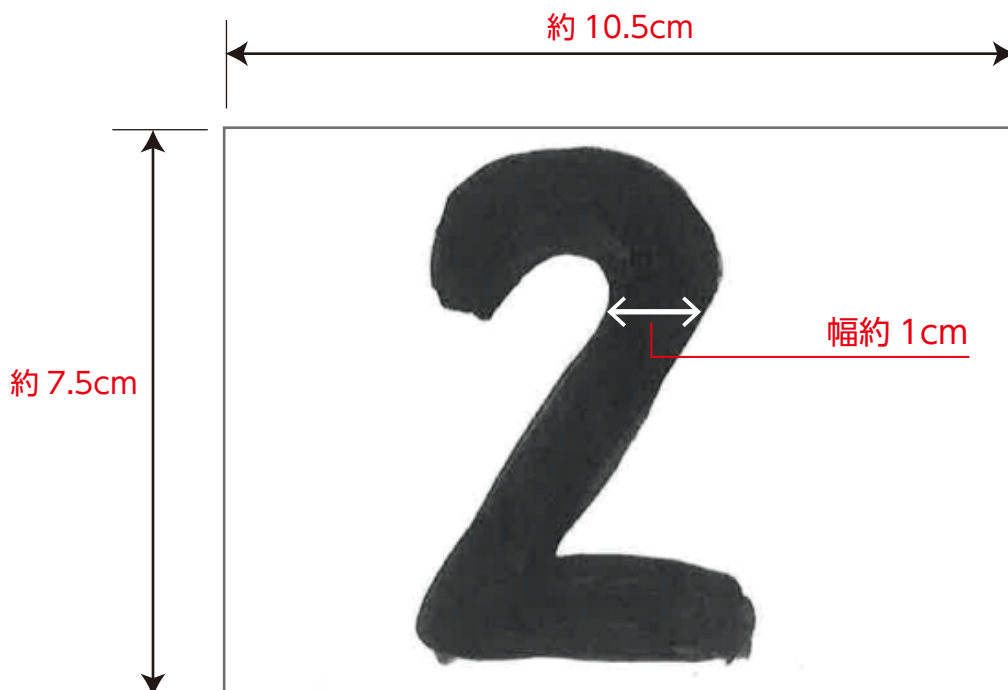
②演習①のtest.pyを削除し、cgi-binフォルダ下に①で作成したdevice2_1.pyを格納します。

今回はWebブラウザを使用しないため、server.pyを起動し、Studuino:bit側のプログラムcap.pyを実行すれば、cap.pyからdevice2_1.pyが呼ばれ、cap.py⇄device2_1.py間でデータの送受信が行われます。



4. 手書き数字の準備

横 10.5cm×縦 7.5cm 程度の白い紙に、黒いマジックペンで0～9の数字を手書きします。その際、手書き数字が紙の中央になるように、1cm 程度の太さの線で太く描いてください。

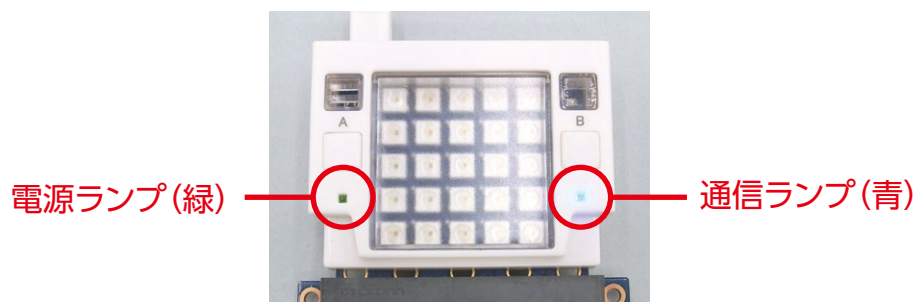


5. 動作確認

- ①sever.batをダブルクリックして、Webサーバを起動します。

```
C:\WINDOWS\system32\cmd.exe
C:\Users\<ユーザー名>\root>python.exe server.py
ポート 80 でWebサーバ稼働中
終了はブラウザを閉じ、Ctrl+Cを押す。
```

- ②Studuino:bitをPCから取り外し、電池ボックスまたはUSBケーブルで給電します。Studuino:bitのリセットボタンを押してcap.pyを実行します。電源ランプ(緑)が点灯している状態で、Bボタンを押してStuduino:bitをWi-Fi接続し、通信ランプ(青)が点灯することを確認してください。



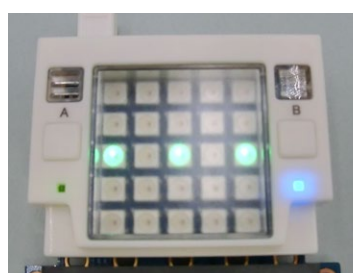
- ③手書き数字を描いた紙を板に貼り、机の上に立てて固定します。
周囲を明るくして小型カメラのレンズを紙に向け、レンズの高さを紙の高さに一致させて机に固定してください。小型カメラと紙の間の距離を12cm程度にします。

- ④Aボタンを押すと、撮影が始まります。
撮影中はStuduino:bitのLEDが1つ赤く光ります。サーバへ画像を送信中はLEDが3つ緑に光ります。



撮影中

```
display.set_pixel(2,2,(10,0,0,))
```



送信中

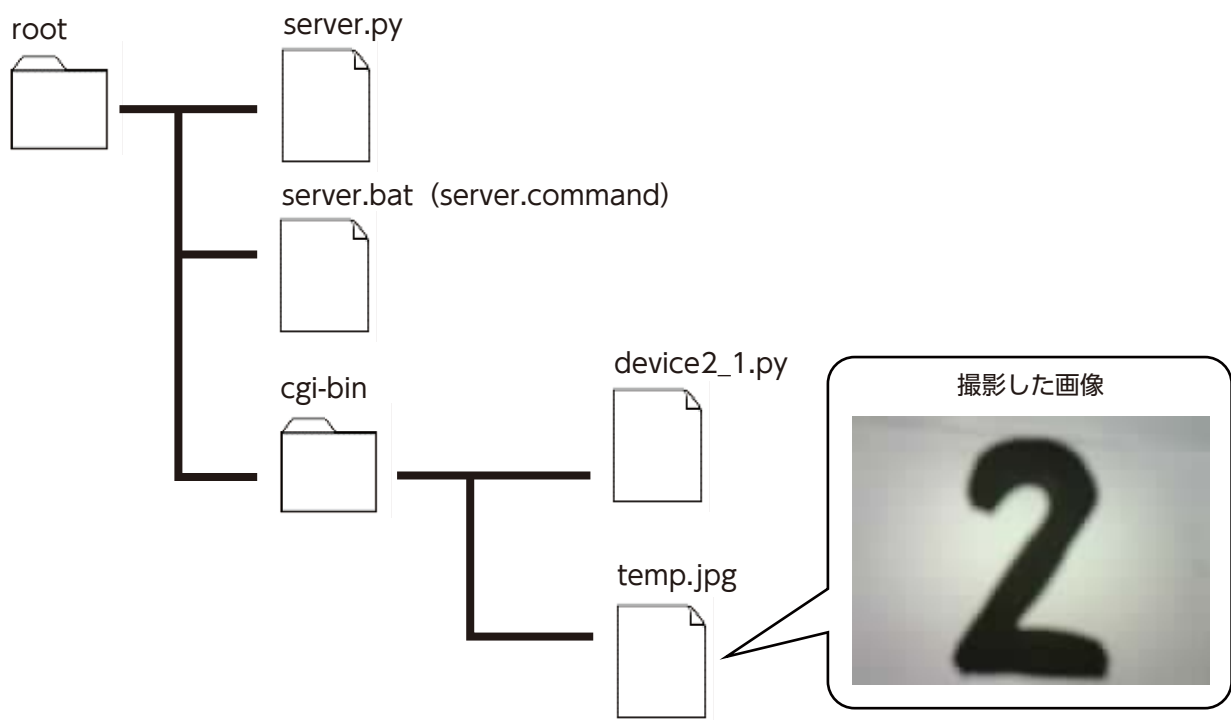
```
display.set_pixel(0,2,(0,10,0,))
display.set_pixel(2,2,(0,10,0,))
display.set_pixel(4,2,(0,10,0,))
```



送信完了

```
display.clear()
```

⑤ cgi-binフォルダ内に撮影されたtemp.jpgが格納されていることを確認してください。



⑥temp.jpgを開いて撮影された画像を確認してください。撮影された数字ができるだけ大きく、画角から外れないように、何度か試行錯誤して紙とカメラの間の上下左右および前後の位置を調整してください。

学習テーマ 3

AI を使った手書き数字認識

[学習内容]

- 人工知能（AI）とはなにか
- 手書き数字認識システム動作の流れ
- ライブラリのインストール
- scikit-learn のデータセットを確認
- 画像の前処理
- 教師データの学習と手書き数字の予測
- 手書き数字認識システムの構築

準備④ 人工知能 (AI) とはなにか

人工知能とは

人工知能というと、映画やアニメに出てくる人間の言葉を話すロボットのようなものをイメージする人も多いかもしれませんが、その適用範囲は広く研究者によって解釈も様々です。

概念的には、「人間が行う知的行動や作業をコンピュータで模倣するシステム」のことを「人工知能」と呼びます。

人工知能はAIとも呼ばれますが、これは「Artificial Intelligence (人工的な知能)」の頭文字を取ったものです。

人工知能の歴史

人工知能の開発は、第二次世界大戦後の1950年頃始まり大きく3つの方式が開発され現代にいたっています。

推論探索システム

最初に開発された人工知能は「推論探索システム」と呼ばれるもので、考えられる選択肢を試行しながら結論を探し出していくという方式です。例えばチェスで打てる手をすべて探索してその中から最良と判断されるものを選択していくという形で答えを見つけ出すものも推論探索システムのひとつです。

この方法では、選択肢が多くなればなるほど答えを見つけ出すのに時間がかかること、当時のコンピュータが今ほど性能が高くなかったこと、用途が限定されることなどから適用範囲はそれほど広くはありませんでした。コンピュータが今ほど普及していなかったことも、その要因のひとつと言えるでしょう。

エキスパートシステム

「推論探索システム」に変わって開発された方式が「エキスパートシステム」と呼ばれるものです。

「エキスパートシステム」とは文字通り「エキスパート(専門家)」のシステムで、ある特定の分野の問題に関する情報とそれに基づく処理の手順を予めシステムとして組み込んだものであり、利用者との対話によって答えを導き出していきます。

例えば、病院に行こうと考えている人が自分の症状について答えていくと、病名を推測し診療科を提示してくれるような問診システムに応用されています。もちろん、このようなシステムで判断できる範囲は限られているため、最終的には医師の診断を受ける必要がありますが、どの科にかかるべきかを事前に判断できるため、より適切な診療科を選択できるようになります。

エキスパートシステムは、「推論エンジン」と呼ばれる処理を行う部分と、判断基準となる情報である「知識」を分けていたため、「知識」の情報を更新することで精度を向上できるという点で「推論探索システム」よりも優れていました。

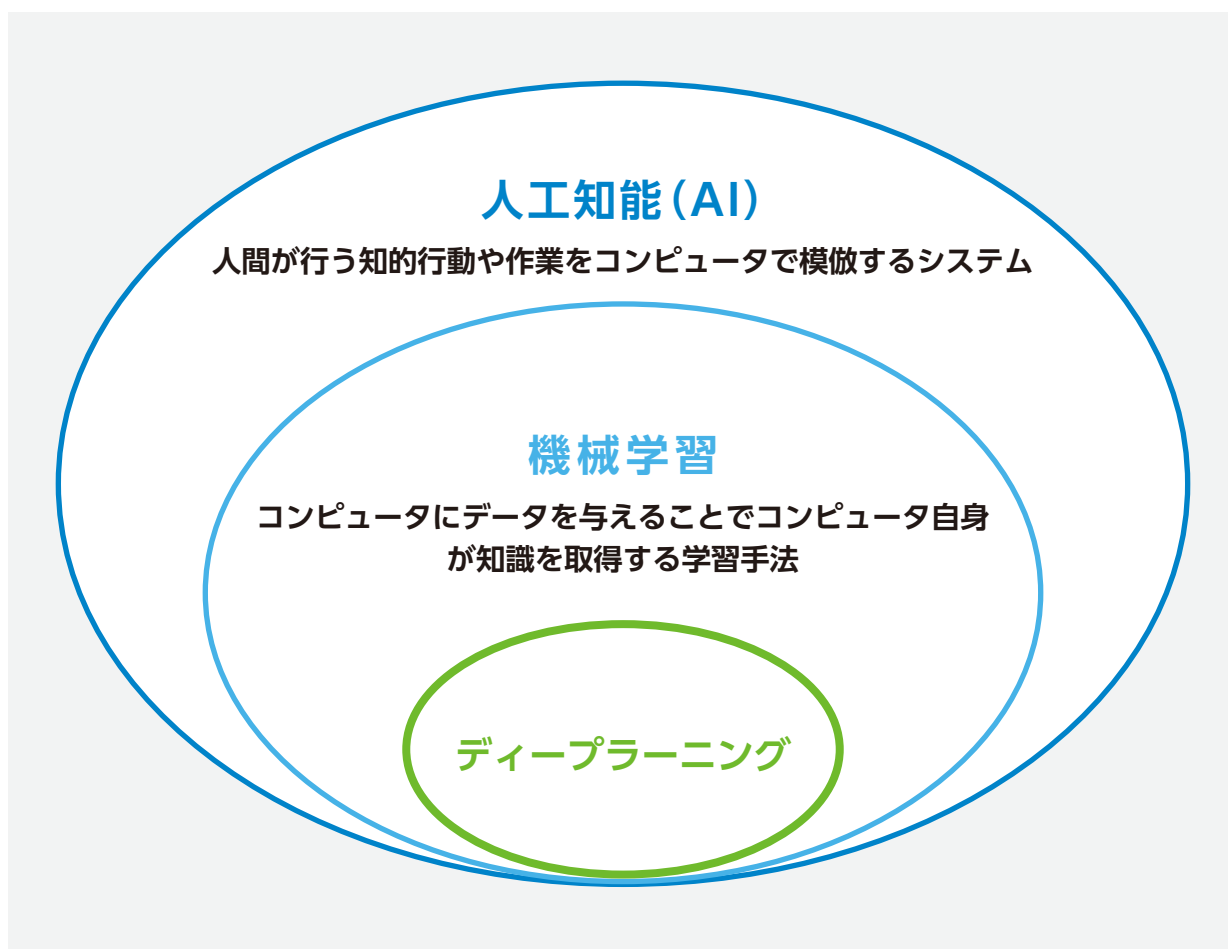
しかしながら、「知識」は人間が更新する必要があること、専門家が考えることを定型化することが難しかったことが課題となっていました。

機械学習

これまでのシステムでは人工知能の精度を向上させるためには人間が知識を与えなければならなかったのに対して、2006年にコンピュータ自身が知識を習得できる「ディープラーニング（深層学習）」が発明され、人工知能の精度は飛躍的に向上することになります。コンピュータ自身が知識を習得する仕組みを「機械学習」と呼び、「ディープラーニング（深層学習）」もそのひとつです。

機械学習の方式は現在多くのシステムで利用されており、例えばスマートフォンの音声認識機能も利用者の発話内容や結果に対する行動を学習して精度が向上します。使えば使うほど精度が向上していくのが機械学習の特長です。

機械学習による人工知能が進化した背景には、機械学習のアルゴリズムの発明はもちろんですが、知識の元となる莫大な情報の流通を可能にしたインターネットの普及や、その情報を処理できる高い計算能力を持ったコンピュータが開発されたことも機械学習の考え方を生み出した背景にあると考えられます。



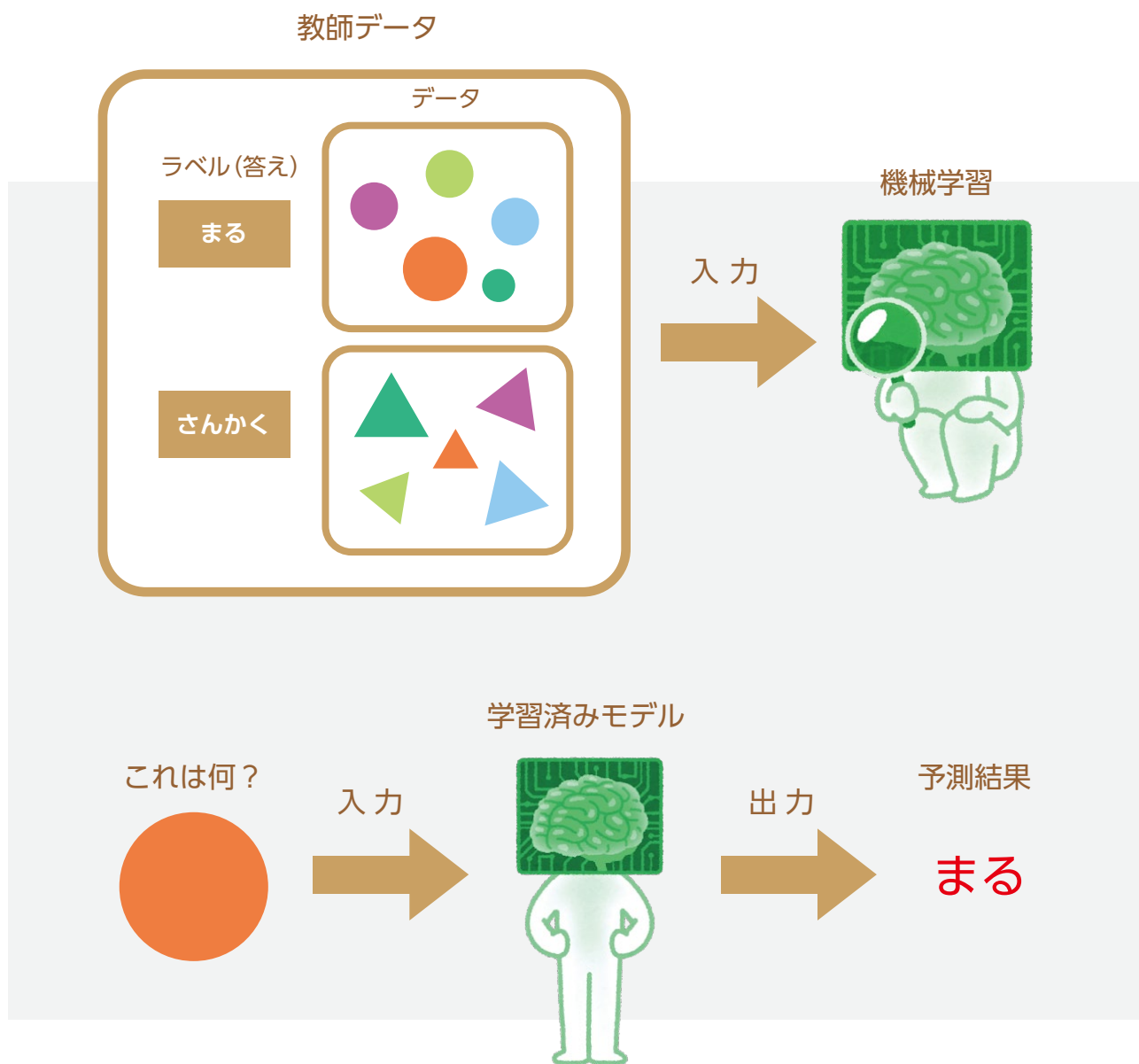
人工知能と機械学習

現代の人工知能の基礎となっている機械学習についてもう少し詳しく見ていきましょう。

機械学習とは、コンピュータがデータを分析して処理するアルゴリズム（情報の処理手順）を搭載したものであることは先に述べた通りです。

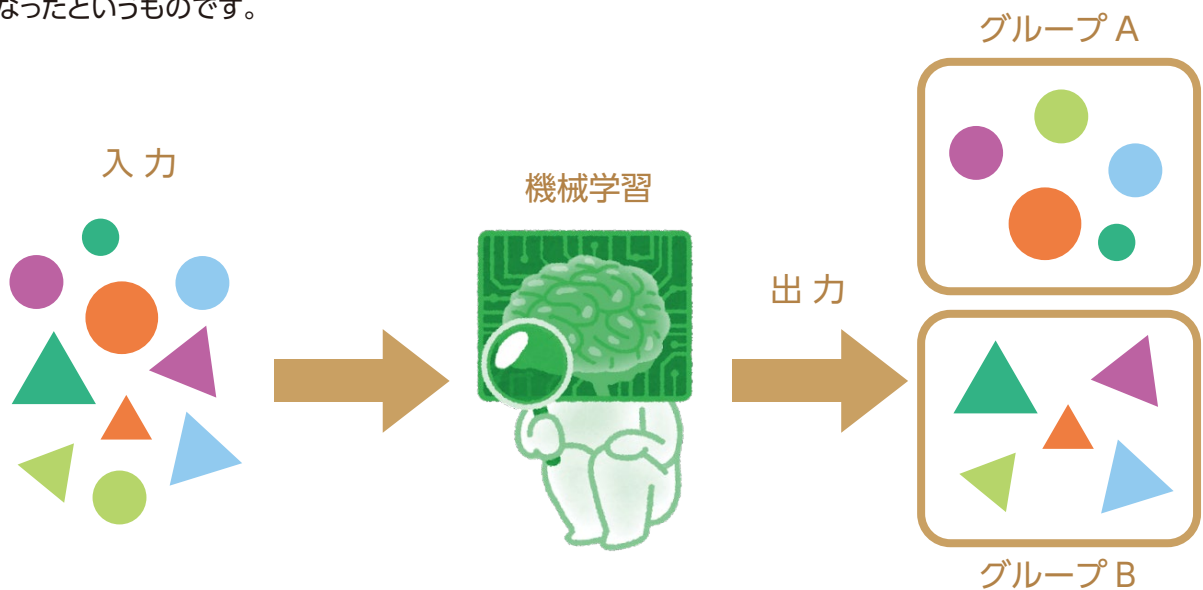
コンピュータが学習する方法として大きく「教師あり学習」と「教師なし学習」に分けられます。

「教師あり学習」とは情報と答えをセットでコンピュータに与える学習方法で、例えば犬の画像とこれが犬であるという情報を一緒にコンピュータに与えることで、コンピュータは他の犬の画像も犬として認識できるようになるものです。



「教師なし学習」とは答えを与えずに情報だけをあたえ、多くの情報をコンピュータが解析、分類を繰り返して自分で答えを見つける学習方式です。

「教師なし学習」の有名な事例として、Google社が2012年に行った「猫の認識」があります。これは、大量の猫の画像を猫という情報を加えずにコンピュータに与えた結果、コンピュータがそれらが猫であると認識できるようになったというものです。



「教師あり学習」は答えが与えられていますので、認識精度は短時間で向上させられる一方、答え付きの情報を人間が用意しなければならないため手間がかかります。

「教師なし学習」は精度向上に時間がかかるものの、データさえあれば人間の手間はほとんどかからないことと、人間が考えなかったことをコンピュータが発見する可能性があります。

もうひとつ、「強化学習」という機械学習の方式もあります。

「強化学習」とは、コンピュータに試行錯誤させて、その達成度に対して評価（報酬）が与えられ、コンピューターはより高い評価を得るために、行動を修正することで目標に近づく学習させる方法です。つまり、ある環境においてコンピュータが現在の状態を観測し、取るべき行動を決定しながら答えを見つけていくという方法で、人間が答えを与えないという意味では「教師なし学習」とも言えますし、一定の基準があるという意味では「教師あり学習」とも言えるかもしれません。

人工知能と私たちの生活

このように進化した人工知能が私たちの生活の中でどのように役立っているのでしょうか？

私たちの身の回りでは、以下のようなところで、人工知能が使われています。

スマートスピーカー



言葉で話しかけると、音楽を再生したり、欲しい情報を音声で教えてくれるスマートスピーカーでは、以下のような機能が人工知能でできています。人間の音声を聞き分けて、音声を文字データに変換する機能、聞き取った文章から、その意味や目的を理解する機能、意味や目的に応じた適切な応答内容を検索する機能、検索結果から応答文章を組み立てる機能、などが人工知能の技術によって実現されています。

顔認識



画面を見るだけでロック解除できるスマートフォンや、SNSにアップロードした写真の中から人の顔を認識し、さらにそれが誰であるかまでも判別する機能に人工知能が働いています。顔の識別に用いる顔の特徴を検出し顔情報として記録する機能、登録済みの顔情報と検出した顔情報を照合し、本人かどうかを判別する機能などが、人工知能で実現されています。

登録時と識別時では、化粧をしている、眼鏡をしている、ケガをしているなど、データが完全に一致しないこともありますが、そのような違いがあっても判別できるのは、人工知能が判断しているからです。

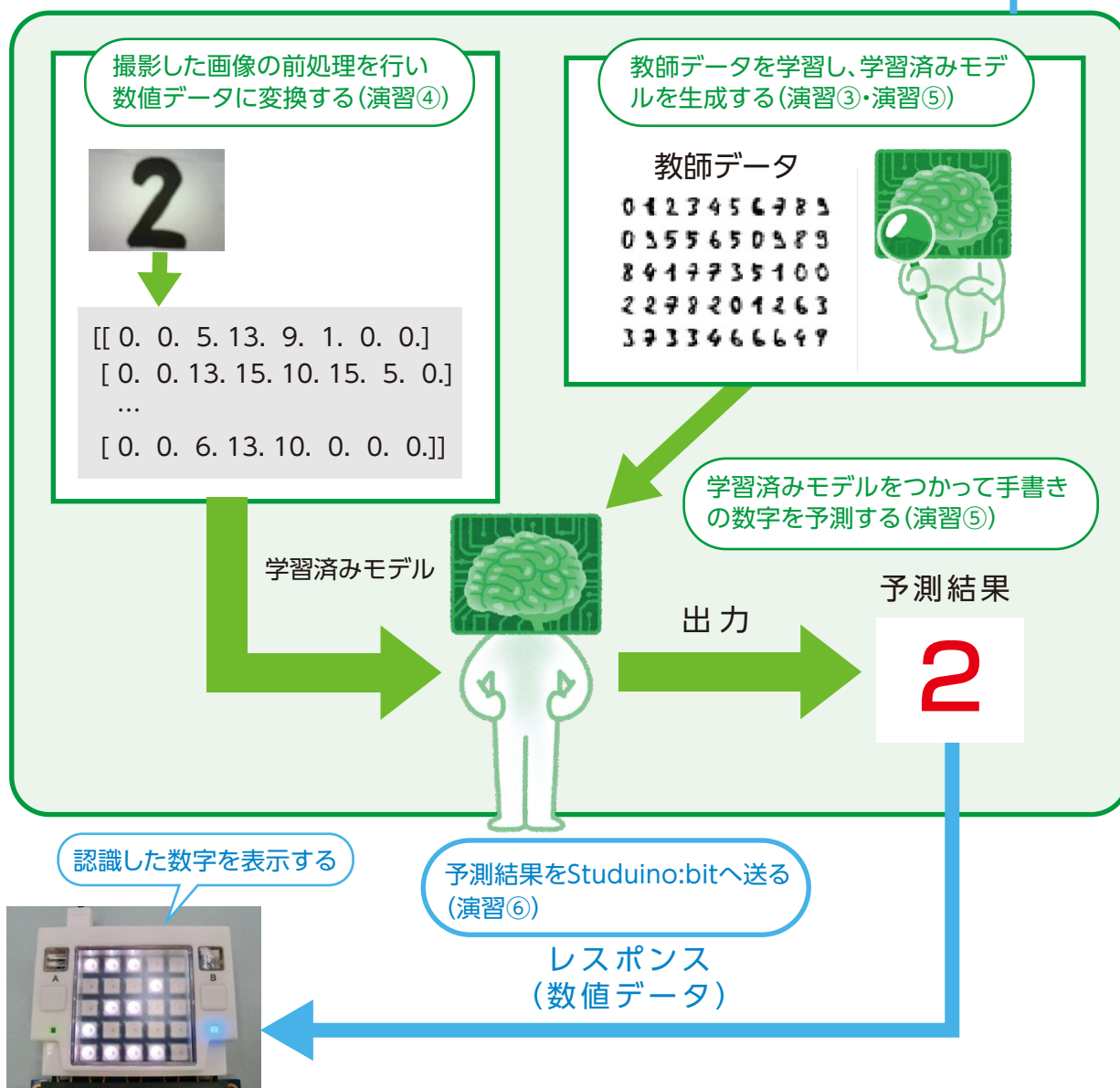
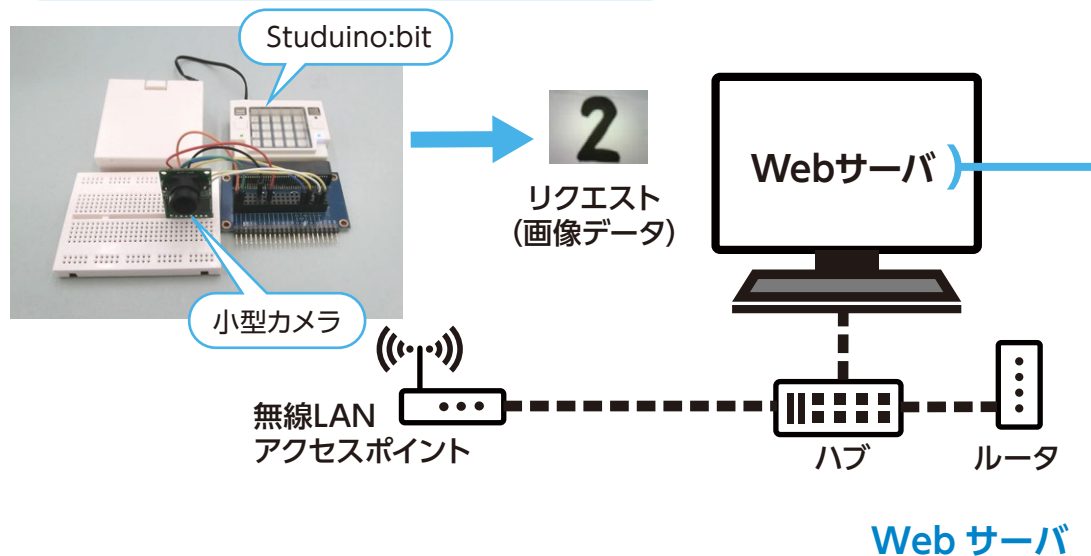
自動運転車



自動車も人工知能の採用が活発な分野のひとつです。完全な自動運転はまだ実用化されていませんが、駐車したい場所を選ぶだけで車が勝手に動いてくれるパーキングアシストは実用化されていますし、ドライバーの表情や声の調子からストレス状況を判断し、安全運転をサポートする研究なども人工知能の技術を応用したものです。

準備⑤ 手書き数字認識システム動作の流れ

Studuino:bitから撮影した画像をPCへ送る(演習②)



準備⑥ ライブラリのインストール

画像の前処理や、数字の学習、予測で使用する以下のライブラリ(モジュール)はPython(Webサーバ)には対応していますが、MicroPython(Studuino:bit)には対応していません。したがって、これらをWebサーバとして使用するPCにインストールします。

使用するライブラリ

scikit-learn (サイキットラーン)	「Python」で記述された機械学習ライブラリであり、機械学習に必要な多数のアルゴリズムが実装されています。関連書籍も多く出版され、オープンソースとして提供されており誰でも無償で利用でき、サンプルとなるデータセットも用意されているため、初めて機械学習を学ぶツールとして最適です。
Pillow (ピロー)	画像処理ライブラリであり、リサイズやトリミング、輝度の変更など画像に関する簡単な処理を行うことができます。 この学習テーマでは手書き数字の画像の処理に使用します。
NumPy (ナムパイ)	多次元配列を扱う数値演算ライブラリであり、機械学習やディープラーニングを行う際に利用されます。演算を高速に処理できるため、機械学習のようなベクトルや行列といった大量のデータを処理する場合に広く利用されています。
Matplotlib (マットプロットリブ)	グラフ描画を行うライブラリです。 今回の学習テーマでは手書き数字の表示(可視化)に使用します。

①外部ライブラリのインストールはコマンドプロンプト(Macの場合はターミナル)を起動して、pipというコマンドを実行して行います。

pip自体も随時更新されているので、以下のコマンドでアップグレードを行います。

Windows の場合

```
pip install --upgrade pip
```

Mac の場合

```
pip3 install --upgrade pip3
```

パスの設定をしなかった場合(Add Python3.* to PATHにチェックを入れなかった場合)は下記のコマンドを実行してください。

Windows の場合

```
python -m pip install --upgrade pip
```

Mac の場合

```
python3 -m pip install --upgrade pip
```

②「scikit-learn」を以下のコマンドでインストールします。

Windows の場合

```
pip install scikit-learn
```

Mac の場合

```
pip3 install scikit-learn
```

③「Pillow」を以下のコマンドでインストールします。

Windows の場合

```
pip install Pillow
```

Mac の場合

```
pip3 install Pillow
```

④「NumPy」を以下のコマンドでインストールします。

Windows の場合

```
pip install numpy
```

Mac の場合

```
pip3 install numpy
```

⑤「Matplotlib」を以下のコマンドでインストールします。

Windows の場合

```
pip install matplotlib
```

Mac の場合

```
pip3 install matplotlib
```

パスの設定をしなかった場合 (Add Python3.* to PATHにチェックを入れなかった場合は下記のコマンドを実行してください。)

Windows の場合

```
python -m pip install scikit-learn  
python -m pip install Pillow  
python -m pip install numpy  
python -m pip install matplotlib
```

Mac の場合

```
python3 -m pip install scikit-learn  
python3 -m pip install Pillow  
python3 -m pip install numpy  
python3 -m pip install matplotlib
```

●ライブラリ名(モジュール名)について

インストール時のライブラリ名は、基本的にインポート時と同じものを使用します。例外として、上記の scikit-learn と Pillow では、インストール時はそれぞれ scikit-learn と Pillow を使用し、インポート時はそれぞれ sklearn と PIL を使用します。

演習③ scikit-learnのデータセットを確認

scikit-learnには手書きの数字のデータセットや、アヤメの品種のデータセット、ワイン認識のデータセット、ボストンの住宅価格のデータセットなどが用意され、教師データとして使用することができます。

本学習テーマでは手書き数字のデータセットを利用します。

1.scikit-learnに用意されている教師データを確認しましょう。

- ① IDLEを使用して下記のsklearn1.pyを作成します。

sklearn1.py

```
from sklearn import datasets

digits = datasets.load_digits()
print(' データ数 : ',len(digits.data))
print(' 画像データの中身 : ',digits.data[0])
print(' 画像データの形 : ',digits.data[0].shape)
print(' ラベル (何の数字か) ',digits.target[0])
```

- ② プログラムを実行すると下記のような結果が表示されます。

```
y
データ数 : 1797
画像データの中身 : [ 0.  0.  5. 13.  9.  1.  0.  0.  0.  0. 13. 15. 10. 15.  5
.  0.  0.  3.
15.  2.  0. 11.  8.  0.  0.  4. 12.  0.  0.  8.  8.  0.  0.  5.  8.  0.
 0.  9.  8.  0.  0.  4. 11.  0.  1. 12.  7.  0.  0.  2. 14.  5. 10. 12.
 0.  0.  0.  0.  6. 13. 10.  0.  0.  0.]
画像データの形 : (64,)
ラベル (何の数字か) 0
>|
```

全部で1797個のデータが含まれています。

dataには手書きの数字の画像を数値化した要素数64の1次元配列(ベクトル)が格納されており、今回は0番目を表示させました。

targetには何の数字かのラベル(こたえ)が格納されています。

target 何の数字か	data 画像を数値化したデータ
0	[0. 0. 5. 13. 9. 1. 0. 0. 0. 0. 13. 15. 10. 15. 5. 0. 0. 3. 15. 2. 0. 11. 8. 0. 0. 4. 12. 0. 0. 8. 8. 0. 0. 5. 8. 0. 0. 9. 8. 0. 0. 4. 11. 0. 1. 12. 7. 0. 0. 2. 14. 5. 10. 12. 0. 0. 0. 0. 6. 13. 10. 0. 0. 0.]
⋮	

要素数64の
1次元配列(ベクトル)

1797 個

2.scikit-learnに用意されている教師データを画像として表示させてみましょう。

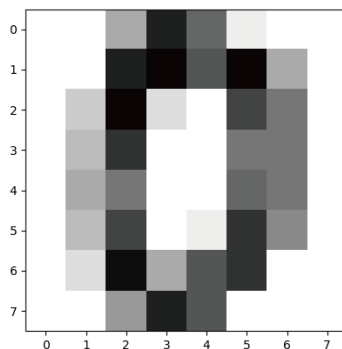
①IDLEを使用して下記のsklearn2.pyを作成します。

sklearn2.py

```
from sklearn import datasets
import matplotlib.pyplot as plt

digits = datasets.load_digits()
print(' ラベル (何の数字か) ', digits.target[0])
digitsReshape = digits.data[0].reshape(8, 8)
plt.imshow(digitsReshape, cmap=plt.cm.gray_r)
plt.show()
```

②プログラムを実行するとshellにラベルが表示され、画像が表示されます。



▶プログラムの解説

```
digits = datasets.load_digits()
```

手書き数字データセットを読み込んで辞書型オブジェクトを返します。
データセットは以下のキーと値を持つ辞書のような型になっています。

キー (辞書名. キー名)	キーに対する値
digits.images	手書き数字画像のピクセルデータが (データ数, 縦幅のピクセル数, 横幅のピクセル数) の形状の 3 次元配列 (1797, 8, 8) を持つ。
digits.data	手書き数字画像のピクセルデータが (データ数, 縦幅のピクセル数 × 横幅のピクセル数) の形状の 2 次元配列 (1797, 64) を持つ。
digits.target	手書き数字画像のラベルデータが (データ数,) の形状の 1 次元配列 (1797,) を持つ。 ラベルデータの 1 次元配列の各要素は各画像に対応するラベル (文字列型) である。

下記のコードで要素数64のベクトル(1次元配列)を8×8の行列(2次元配列)に変換しています。
 digits.dataは2次元配列ですが、そのうちの0番目のdigits.data[0]は1次元配列になります。

```
digitsReshape = digits.data[0].reshape(8, 8)
```

1次元配列の画像データ reshape(変換後の行数、変換後の列数)

digits.data[0]

0	0	5	13	9	...	13	10	0	0	0
---	---	---	----	---	-----	----	----	---	---	---

64



reshape(8,8)

digitsReshape

8

0	0	5	13	9	1	0	0
0	0	13	15	10	15	5	0
⋮							
0	2	14	5	10	12	0	0
0	0	6	13	10	0	0	0

8

scikit-learn の手書き数字のデータセットには白に 0、黒に 16 が割り当てられています。
 下記のコードで数値に色を割り当てて、画像化しています。

```
plt.imshow(digitsReshape, cmap=plt.cm.gray_r)
```

plt.show()

gray_rはグレースケールを逆転したものであり、画素値が小さいほど白く、画素値が大きいほど黒く、白黒が表されます。

0	0	5	13	9	1	0	0	← 数値が小さい場合、白に近い色
0	0	13	15	10	15	5	0	← 数値が大きい場合、黒に近い色
0	3	15	2	0	11	8	0	
0	4	12	0	0	8	8	0	
0	5	8	0	0	9	8	0	
0	4	11	0	1	12	7	0	
0	2	14	5	10	12	0	0	
0	0	6	13	10	0	0	0	

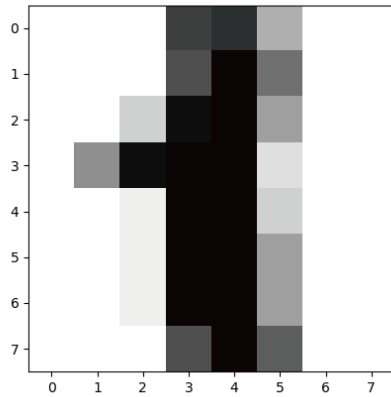
③ 0 番目以外のデータも表示させてみましょう。

sklean2.py の赤字部分の 0 を 1,2,3... と打ち換えてプログラムを実行してください。

付録 F に scikit-learn の手書き数字データセットを画像化して掲載しています。

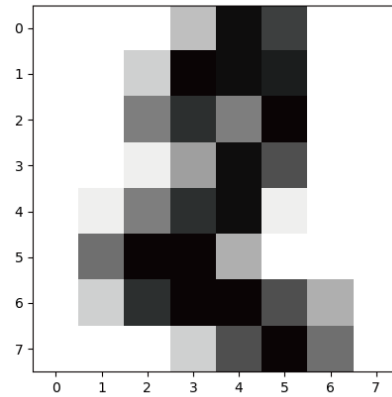
1 番目のデータ

画像ラベル「1」



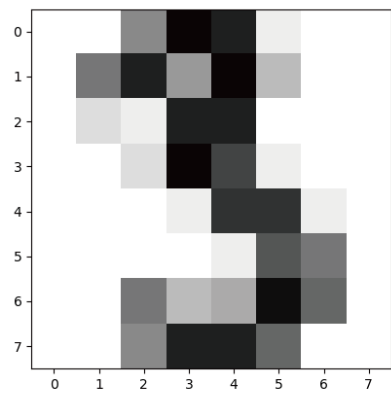
2 番目のデータ

画像ラベル「2」



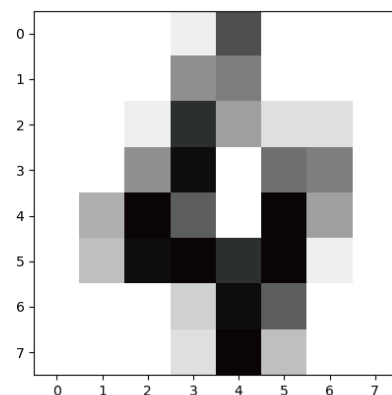
3 番目のデータ

画像ラベル「3」



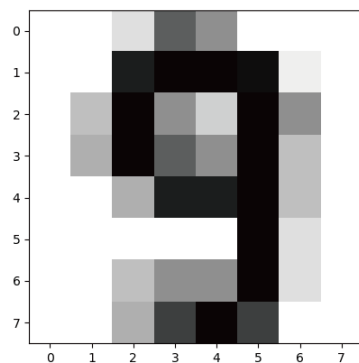
4 番目のデータ

画像ラベル「4」



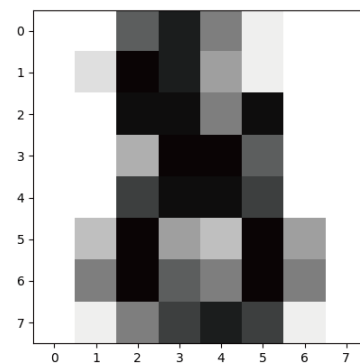
1795 番目のデータ

画像ラベル「9」



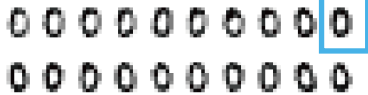
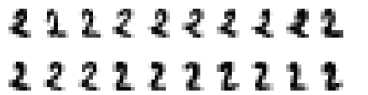
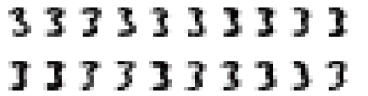
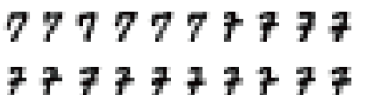
1796 番目のデータ

画像ラベル「8」



scikit-learn のデータセットは下記のようなデータ構造であることがわかります。

- 0 ～ 9 までの手書きの数字が 1797 個用意されている。
- 1 つの画像は要素数 64 (8×8) の 1 次元配列 (ベクトル) で、色がグレースケール (白 0 ～ 黒 16) で表現されている。

教師データ 1797 個	
ラベル (答え) target	データ data
0	 
1	
2	
3	
4	
5	
6	
7	
8	
9	

```
[0. 0. 5. 13. 9. 1. 0. 0.
0. 0. 13. 15. 10. 15. 5. 0.
0. 3. 15. 2. 0. 11. 8. 0.
0. 4. 12. 0. 0. 8. 8. 0.
0. 5. 8. 0. 0. 9. 8. 0.
0. 4. 11. 0. 1. 12. 7. 0.
0. 2. 14. 5. 10. 12. 0. 0.
0. 0. 6. 13. 10. 0. 0. 0.]
```

演習⑤では、これらの教師データを使って機械学習を行います。

④imshow() に指定した colormap を gray にして、①～③と同様にデータを表示させてみます。

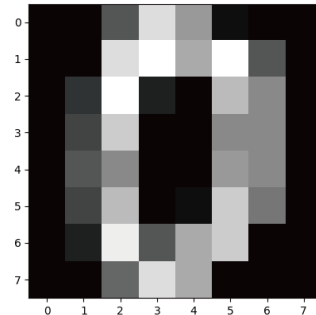
IDLE を使用して下記の sklearn3.py を作成し、実行します。

赤字部分の 0 を 1,2,3... と打ち換えてプログラムを実行し、0 番目以外のデータも表示させてください。

sklearn3.py

```
from sklearn import datasets
import matplotlib.pyplot as plt

digits = datasets.load_digits()
print(' ラベル (何の数字か) ', digits.target[0])
digitsReshape = digits.data[0].reshape(8, 8)
plt.imshow(digitsReshape, cmap=plt.cm.gray)
plt.show()
```



▶プログラムの解説

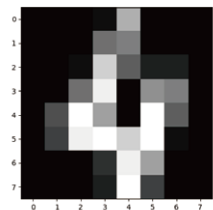
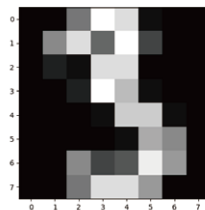
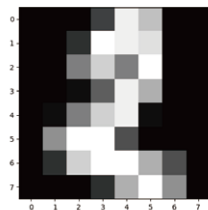
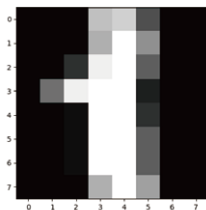
plt.imshow(Ⓐ, Ⓑ)

Ⓐで指定された配列または Image オブジェクトの要素の値をⒷで指定した colormap に従った色に変換した画像を描画します。

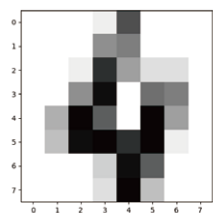
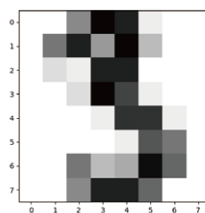
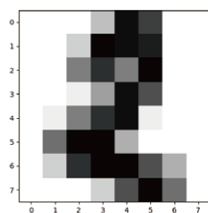
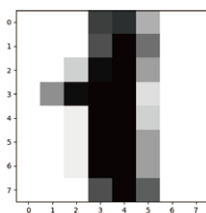
ここで指定した gray はグレースケールであり、画素値が小さいほど黒く、画素値が大きいほど白く白黒が表されます。gray_r はグレースケールを逆転したものであり、画素値が小さいほど白く、画素値が大きいほど黒く白黒が表されます。

imshow の cmap(colormap) 引数

gray



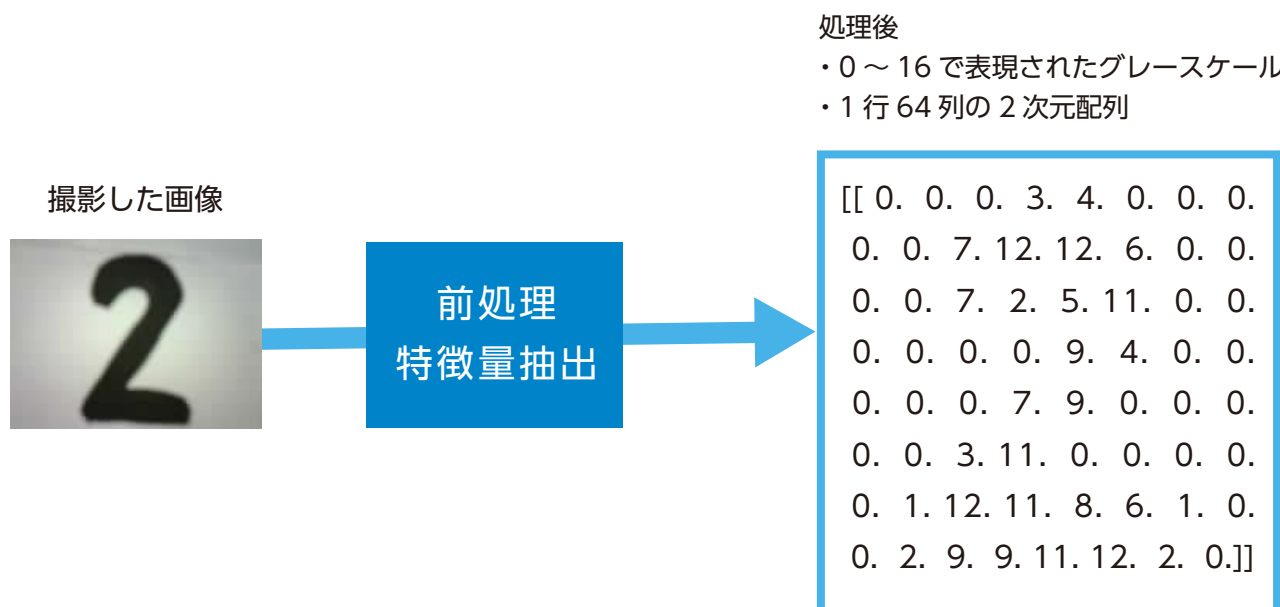
gray_r



演習④ 画像の前処理

撮影した画像から文字を予測するために、演習③で確認した教師データと同じ0～16で表現されたグレースケールになるよう、画像に前処理を行って、特徴量を抽出します。

また、教師データは要素数64の1次元配列ですが、演習⑤で扱うscikit-learnの機械学習メソッドpredict()は1次元配列ではなく2次元配列を引数にとるため、入力する画像は1行64列の2次元配列へ変換します。



①学習テーマ2演習②で撮影した画像をmatplotlibのグラフ上に表示させます。

IDLEを使用して下記のimage1.pyのプログラムを作成し、学習テーマ2演習②で撮影した画像ファイルtemp.jpgと同じフォルダに格納してください。

プログラムを実行し、画像が表示されることを確認します。

```
image1.py  
  
import matplotlib.pyplot as plt  
from PIL import Image  
filename = 'temp.jpg'  
im = Image.open(filename)  
plt.imshow(im)  
plt.show()
```



▶プログラムの解説

```
im = Image.open('A' )
```

①: ファイル名またはファイルへのパスを指定します。

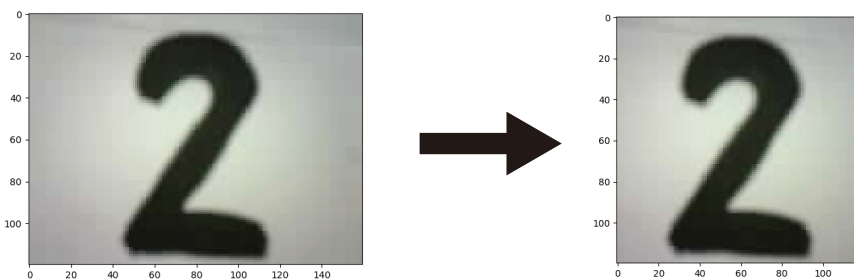
②: ①で指定された画像ファイルの Image オブジェクトを返します。

②撮影した画像を正方形に切り抜きます。

IDLE を使用して下記の image2.py のプログラムを作成し、実行します。

image2.py

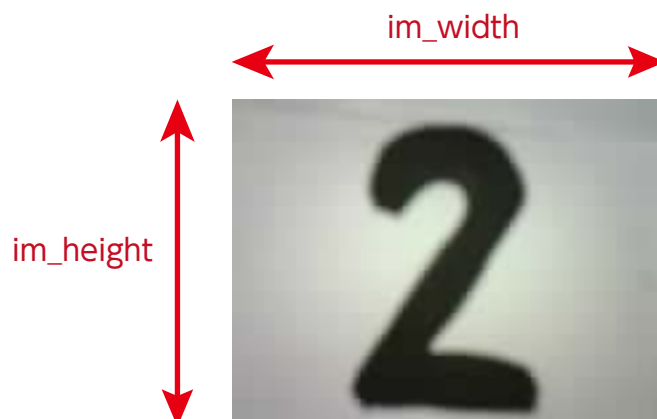
```
import matplotlib.pyplot as plt
from PIL import Image
filename = 'temp.jpg'
im = Image.open(filename)
im_width, im_height = im.size
square_len = min(im.size)
im = im.crop(((im_width - square_len) // 2, (im_height - square_len) // 2,
              (im_width + square_len) // 2, (im_height + square_len) // 2))
plt.imshow(im)
plt.show()
```



▶プログラムの解説

```
im_width, im_height = im.size
```

画像 im のサイズ（幅・高さ）を取得し、幅を im_width に、高さを im_height に代入します。



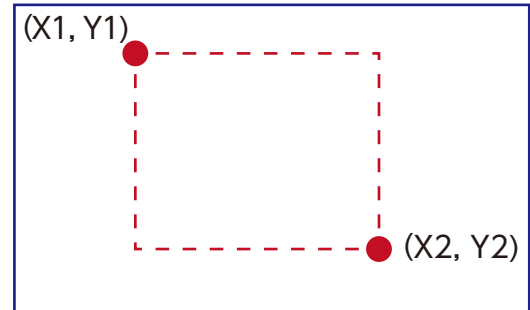
```
square_len = min(im.size)
```

画像 im の幅と高さのうち、短い方を square_len に代入します。

```
im = im.crop((X1, Y1, X2, Y2))
```

(X1, Y1, X2, Y2) で指定した長方形で切り取った画像 (Image オブジェクト) を返します。

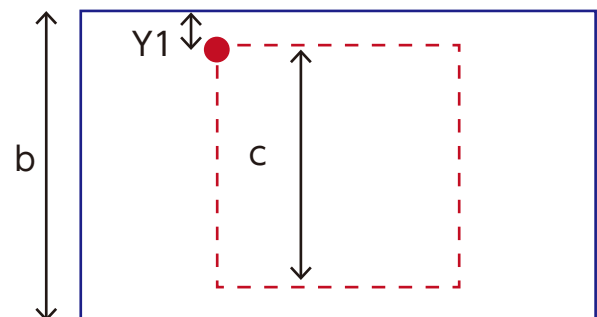
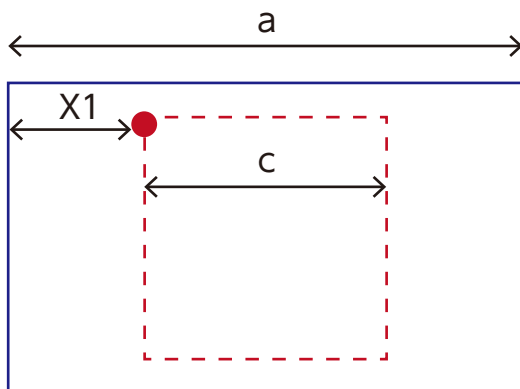
切り取る範囲を (左上の x 座標, 左上の y 座標, 右下の x 座標, 右下の y 座標) というタプルで指定します。



今回は長方形 (画像) の中心から正方形を切り取ります。

$$X1 = (a-c)/2 = (im_width - square_len) / 2$$

$$Y1 = (b-c)/2 = (im_height - square_len) / 2$$



$$\begin{aligned} X2 &= d+c = (a-c)/2 + c \\ &= (a+c)/2 = (im_width + square_len) / 2 \end{aligned}$$

$$\begin{aligned} Y2 &= e+c = (b-c)/2 + c \\ &= (b+c)/2 = (im_height + square_len) / 2 \end{aligned}$$

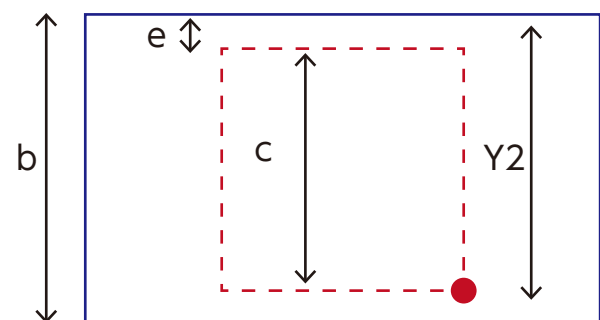
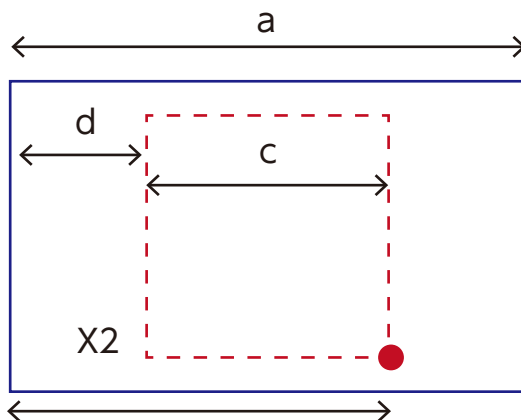
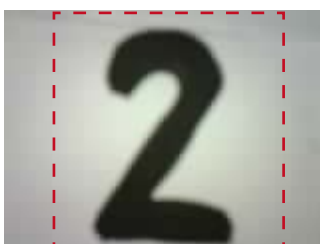


image2.py を実行すると square_len に im_height が反映され、下記の赤い点線部分が切り抜かれます。



なお、プログラムの中の「//」は、整数除算 (切り捨て除算) の算術演算子です。

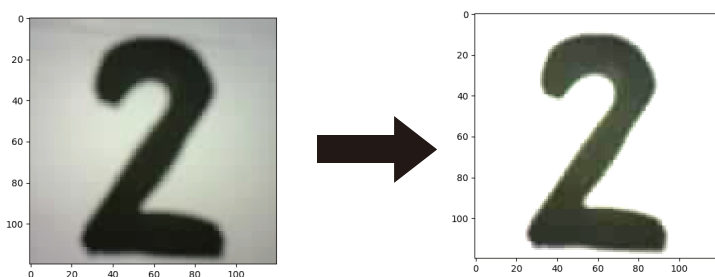
`print(5/2) ⇒ 2.5`

`print(5//2) ⇒ 2`

- ③画像の明暗をはっきりさせるため、明瞭化の処理を追加します。IDLEを使用して下記のimage3.pyのプログラムを作成し、実行します。

image3.py

```
import matplotlib.pyplot as plt
from PIL import Image, ImageEnhance
filename = 'temp.jpg'
im = Image.open(filename)
im_width, im_height = im.size
square_len = min(im.size)
im = im.crop(((im_width - square_len) // 2, (im_height - square_len) // 2,
              (im_width + square_len) // 2, (im_height + square_len) // 2))
im = ImageEnhance.Brightness(im).enhance(2.0)
plt.imshow(im)
plt.show()
```



▶プログラムの解説

```
im = ImageEnhance.Brightness(im).enhance(Ⓐ)
```

- Ⓐ: 輝度の拡張係数（1 より大きい場合は明るく、1 より小さい場合は暗く調整できる）を指定します。

- ④グレースケールによる白黒に変換します。IDLE を使用して下記のimage4.pyのプログラムを作成し、実行します。

image4.py

```
import matplotlib.pyplot as plt
from PIL import Image, ImageEnhance
filename = 'temp.jpg'
im = Image.open(filename)
im_width, im_height = im.size
square_len = min(im.size)
im = im.crop(((im_width - square_len) // 2, (im_height - square_len) // 2,
              (im_width + square_len) // 2, (im_height + square_len) // 2))
im = ImageEnhance.Brightness(im).enhance(2.0)
im = im.convert('L')
plt.imshow(im, cmap="gray")
plt.show()
```

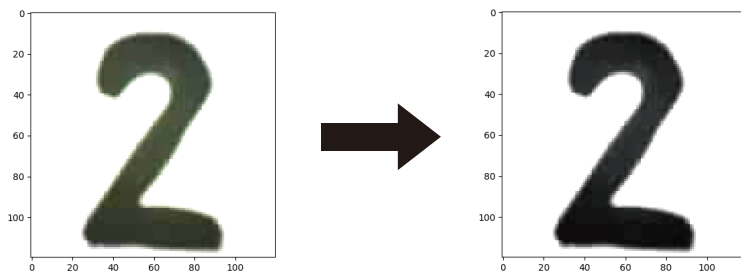
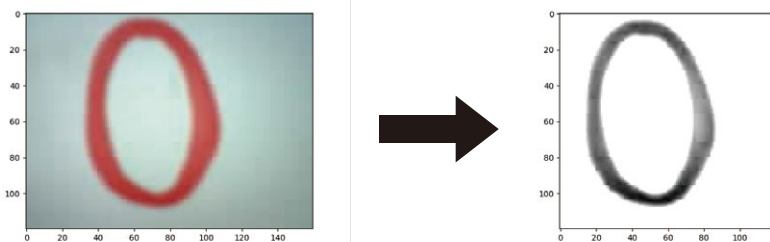


image1.pyとimage4.pyの3行目のtemp.jpgをtemp0.jpg (赤ペンで書いた文字)に変更します。
下記は赤ペンで書いた文字をimage1.pyで表示させたもの(左)と、image4.pyでグレースケール化したもの(右)です。



▶プログラムの解説

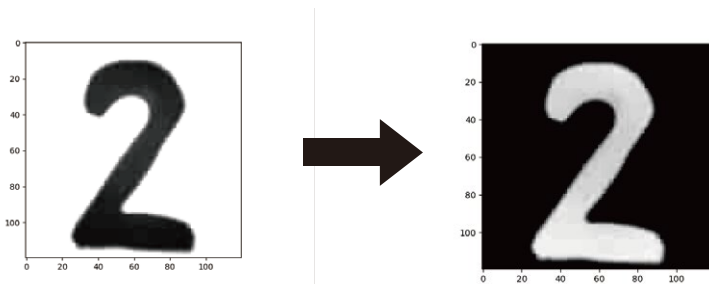
```
im = im.convert('A')
```

Ⓐ: 画像のモードを指定します。グレースケールはLです。

⑤白黒を反転します。IDLEを使用して下記のimage5.pyのプログラムを作成し、実行します。

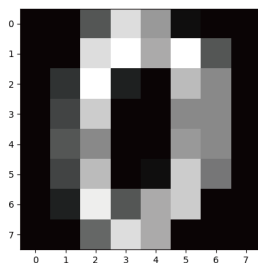
image5.py

```
import matplotlib.pyplot as plt
from PIL import Image, ImageEnhance, ImageOps
filename = 'temp.jpg'
im = Image.open(filename)
im_width, im_height = im.size
square_len = min(im.size)
im = im.crop(((im_width - square_len) // 2, (im_height - square_len) // 2,
              (im_width + square_len) // 2, (im_height + square_len) // 2))
im = ImageEnhance.Brightness(im).enhance(2.0)
im = im.convert('L')
im = ImageOps.invert(im)
plt.imshow(im, cmap="gray")
plt.show()
```



演習③でscikit-learnのデータセットを画像表示させたように、cmap=grayの場合、数字の線の部分が白、背景が黒になります。

scikit-learn の手書き数字



▶プログラムの解説

```
im = ImageOps.invert(Ⓐ)
```

Ⓐで指定された Image オブジェクトの各ピクセルの画素値を反転した Image オブジェクトを返します。

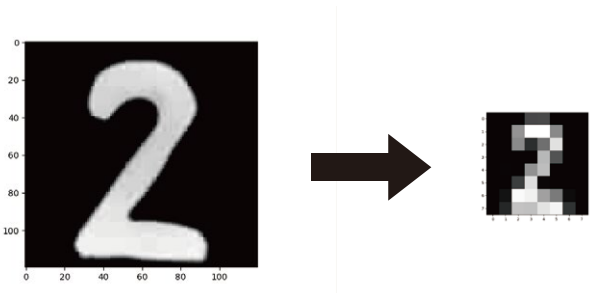
⑥横 8 ピクセル × 縦 8 ピクセルのサイズに画像データを変換します。

IDLE を使用して下記の image6.py のプログラムを作成し、実行します。

image6.py ~ image9.py や image.py が正常に動作しない場合は、コマンドプロンプト (Mac の場合はターミナル) から `pip install --upgrade Pillow` (Mac の場合は `pip3 install --upgrade Pillow`) で Pillow (PIL) をアップグレードしてください。

image6.py

```
import matplotlib.pyplot as plt
from PIL import Image, ImageEnhance, ImageOps
filename = 'temp.jpg'
im = Image.open(filename)
im_width, im_height = im.size
square_len = min(im.size)
im = im.crop(((im_width - square_len) // 2, (im_height - square_len) // 2,
              (im_width + square_len) // 2, (im_height + square_len) // 2))
im = ImageEnhance.Brightness(im).enhance(2.0)
im = im.convert('L')
im = ImageOps.invert(im)
im = im.resize((8, 8), Image.Resampling.LANCZOS)
plt.imshow(im, cmap="gray")
plt.show()
```



▶プログラムの解説

```
im = im.resize(Ⓐ, Ⓑ)
```

Ⓐ: (横幅ピクセル数, 縦幅のピクセル数)のタプル

Ⓑ: リサンプリングフィルタ

Ⓑで指定されたフィルタを用いてⒶで指定されたサイズにしたImageオブジェクトを返します。

リサンプリングフィルタはいくつか種類がありますが、LANCZOSが最も品質が優れています (その分パフォーマンスが悪い)。リサンプリングフィルタを指定しない場合はデフォルトでNEARESTが指定されます (品質は悪いがパフォーマンスが良い)。

⑦画像を2次元Numpy配列に変換します。

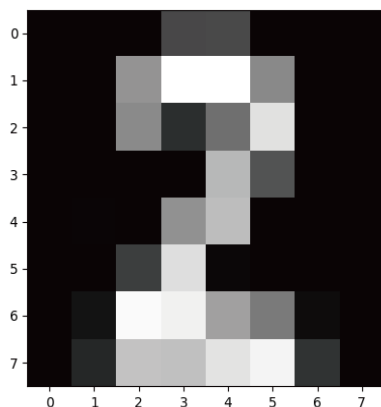
IDLEを使用して下記のimage7.pyのプログラムを作成し、実行します。

image7.py

```
import numpy as np
from PIL import Image, ImageEnhance, ImageOps
filename = 'temp.jpg'
im = Image.open(filename)
im_width, im_height = im.size
square_len = min(im.size)
im = im.crop(((im_width - square_len) // 2, (im_height - square_len) // 2,
              (im_width + square_len) // 2, (im_height + square_len) // 2))
im = ImageEnhance.Brightness(im).enhance(2.0)
im = im.convert('L')
im = ImageOps.invert(im)
im = im.resize((8, 8), Image.Resampling.LANCZOS)
digit = np.asarray(im)
print('digit', digit)
```

下記のように 2 次元配列が出力されます。

```
digit [[ 0  0  1 55 56  0  0  0]
 [ 2  0 112 193 192 103  0  2]
 [ 1  0 104  35  84 170  0  1]
 [ 0  1  0  0 139  63  0  2]
 [ 0  3  0 110 143  0  2  0]
 [ 2  0 47 168  6  0  0  0]
 [ 1 17 188 182 122 93 10  1]
 [ 0 31 147 145 172 185 39  0]]
```



0	0	1	55	56	0	0	0
2	0	112	193	192	103	0	2
1	0	104	35	84	170	0	2
0	1	0	0	139	63	0	1
0	3	0	110	143	0	2	0
2	0	47	168	6	0	0	0
1	17	188	182	122	93	10	1
0	31	147	145	172	185	39	0

▶プログラムの解説

```
digit = np.asarray(Ⓐ)
```

Ⓐで指定された Image オブジェクトについて画素値を要素とした 2 次元 Numpy 配列に変換して返します。

⑧グレースケール化された画像は0～255の数値で表現されていますが、演習③で確認したようにscikit-learnの手書き数字の画像は0～16で表現されているため、画像の数値の範囲を縮小します。

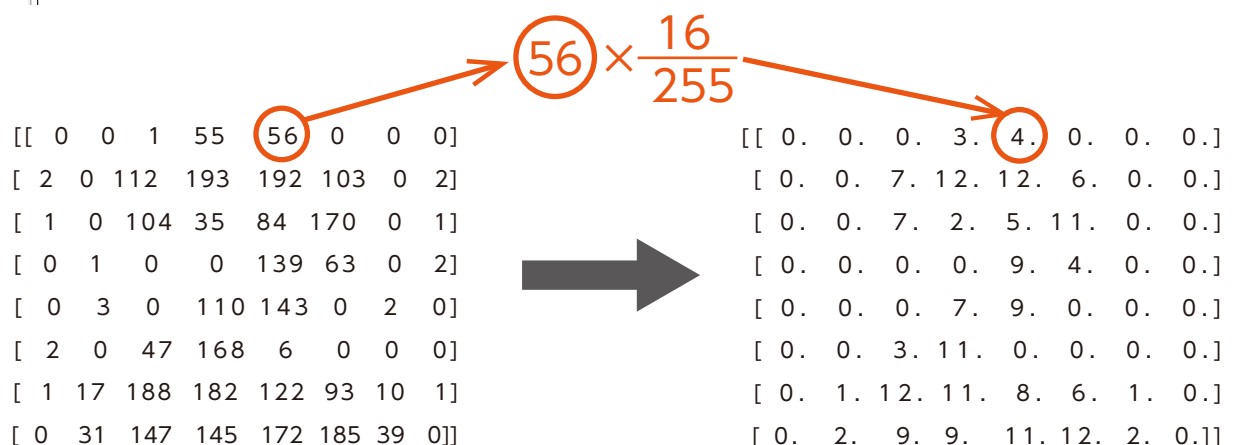
IDLEを使用して下記のimage8.pyのプログラムを作成し、実行します。

image8.py

```
import numpy as np
from PIL import Image, ImageEnhance, ImageOps
filename = 'temp.jpg'
im = Image.open(filename)
im_width, im_height = im.size
square_len = min(im.size)
im = im.crop(((im_width - square_len) // 2, (im_height - square_len) // 2,
              (im_width + square_len) // 2, (im_height + square_len) // 2))
im = ImageEnhance.Brightness(im).enhance(2.0)
im = im.convert('L')
im = ImageOps.invert(im)
im = im.resize((8, 8), Image.Resampling.LANCZOS)
digit = np.asarray(im)
digit = np.round(16*(digit/255))
print('digit', digit)
```

下記のように 2 次元配列が出力されます。

```
digit [[ 0.  0.  0.  3.  4.  0.  0.  0.]
 [ 0.  0.  7. 12. 12.  6.  0.  0.]
 [ 0.  0.  7.  2.  5. 11.  0.  0.]
 [ 0.  0.  0.  0.  9.  4.  0.  0.]
 [ 0.  0.  0.  7.  9.  0.  0.  0.]
 [ 0.  0.  3. 11.  0.  0.  0.  0.]
 [ 0.  1. 12. 11.  8.  6.  1.  0.]
 [ 0.  2.  9.  9. 11. 12.  2.  0.]]
```



▶プログラムの解説

round(A)

Ⓐで指定された配列の各要素を小数点以下で四捨五入して整数値にした新たな配列を返します。

⑨(8,8)から(1,64)の形状の行列(2次元配列)に変換します。

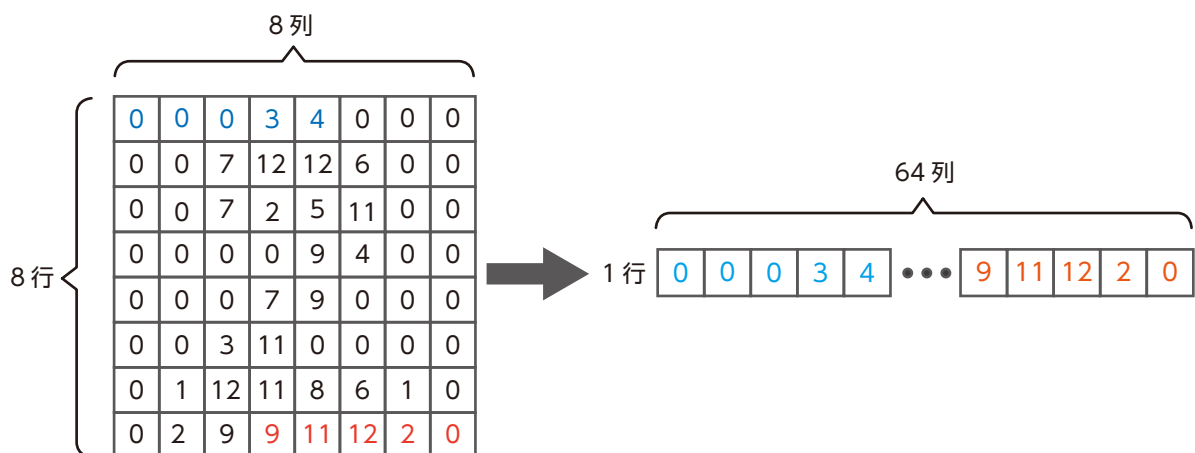
IDLEを使用して下記のimage9.pyのプログラムを作成し、実行します。

image9.py

```
import numpy as np
from PIL import Image, ImageEnhance, ImageOps
filename = 'temp.jpg'
im = Image.open(filename)
im_width, im_height = im.size
square_len = min(im.size)
im = im.crop(((im_width - square_len) // 2, (im_height - square_len) // 2,
              (im_width + square_len) // 2, (im_height + square_len) // 2))
im = ImageEnhance.Brightness(im).enhance(2.0)
im = im.convert('L')
im = ImageOps.invert(im)
im = im.resize((8, 8), Image.Resampling.LANCZOS)
digit = np.asarray(im)
digit = np.round(16*(digit/255))
digit = digit.reshape(1, -1)
print('digit', digit)
```

下記のように (1,64) の 2 次元配列が出力されます。

```
digit [[ 0.  0.  0.  3.  4.  0.  0.  0.  0.  0.  7. 12. 12.  6.  0.  0.  0.  0.
        7.  2.  5. 11.  0.  0.  0.  0.  0.  0.  9.  4.  0.  0.  0.  0.  0.  7.
        9.  0.  0.  0.  0.  0.  3. 11.  0.  0.  0.  0.  1. 12. 11.  8.  6.
        1.  0.  0.  2.  9.  9. 11. 12.  2.  0.]]
```



▶プログラムの解説

```
reshape(1, -1)
```

reshape(1, -1)の引数(-1)のところにはもとの形状から推測された値が入ります。したがって、reshape(1, -1)はreshape(1, 64)と同じです。

- ⑩以降の演習で手書き数字を予測させる際に、ここまで行った画像の前処理、特徴抽出を行います。
処理を関数化しておきます。
下記の image.py を作成して保存してください。

image.py

```
import numpy as np
from PIL import Image, ImageOps, ImageEnhance
def imageToData(filename):
    im = Image.open(filename)
    im_width, im_height = im.size
    square_len = min(im.size)
    im = im.crop(((im_width - square_len) // 2, (im_height - square_len) // 2,
    (im_width + square_len) // 2, (im_height + square_len) // 2))
    im = ImageEnhance.Brightness(im).enhance(2.0)
    im = im.convert('L')
    im = ImageOps.invert(im)
    im = im.resize((8, 8), Image.Resampling.LANCZOS)
    digit = np.asarray(im)
    digit = np.round(16*(digit/255))
    digit = digit.reshape(1, -1)
    return digit
```

Mac 環境のみ先頭行に以下のコードを追加してください。

```
#!/usr/bin/env python3
```

演習⑤

教師データの学習と手書き数字の予測

scikit-learn から提供されている手書き数字のデータセットを機械学習し、自分で書いた手書きの数字を予測します。

1. 機械学習

scikit-learn の教師データ

(演習③で確認済み)

```
0 1 2 3 4 5 6 7 8 9
0 3 5 5 6 5 0 3 8 9
8 4 1 7 7 3 5 1 0 0
2 2 7 8 2 0 1 2 6 3
3 7 3 3 4 6 6 6 4 9
```



2. 予測

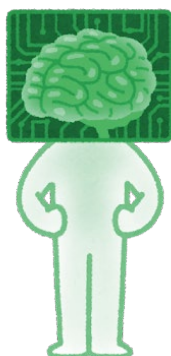
手書きの画像から特徴抽出
(演習④で作成したプログラム
image.py を使用)



入力



学習済みの分類モデル



出力



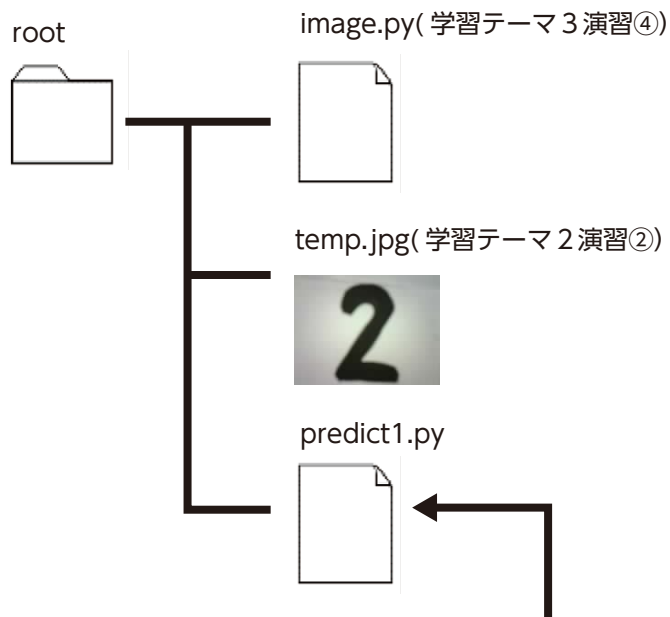
予測結果

2

```
[[ 0.  0.  0.  3.  4.  0.  0.  0.
  0.  0.  7. 12. 12.  6.  0.  0.
  0.  0.  7.  2.  5. 11.  0.  0.
  0.  0.  0.  0.  9.  4.  0.  0.
  0.  0.  0.  7.  9.  0.  0.  0.
  0.  0.  3. 11.  0.  0.  0.  0.
  0.  1. 12. 11.  8.  6.  1.  0.
  0.  2.  9.  9. 11. 12.  2.  0.]]
```

※「分類モデル」とは入力したデータに対する出力
(予測結果) を導き出す仕組みのことです。

①IDLE を使用して下記の predict1.py のプログラムを作成し、演習②で撮影した画像ファイル temp.jpg と演習④で作成した image.py と同じフォルダに格納してください。



```
from sklearn.linear_model import LogisticRegression
from sklearn import datasets
import image
ftemp = 'temp.jpg'
digit = image.imageToData(ftemp)
digits = datasets.load_digits()
X = digits.data
y = digits.target
model = LogisticRegression(solver='liblinear')
model.fit(X, y)
digit_pred = model.predict(digit)
print(digit_pred)
```

※digits は scikit-learn の手書き数字のデータセットで、digit はユーザの手書き数字データです。

②プログラムを実行して、予測結果が表示されることを確認します。

| [2]

▶プログラムの解説

```
from sklearn.linear_model import LogisticRegression
```

from パッケージ名 . モジュール名 import クラス名

複数のモジュールをまとめたものがパッケージです。

上記のプログラムで sklearn パッケージの linear_model モジュールから LogisticRegression (ロジスティック回帰) クラスをインポートしています。

```
model = LogisticRegression(solver='liblinear')
```

LogisticRegression クラスからオブジェクト model を生成（インスタンス化）しています。

オブジェクト名 = クラス名（引数）

liblinear は liblinear は比較的小さいデータセットに対して用いる solver（収束計算方法）です。

scikit-learn には LogisticRegression（ロジスティック回帰）の他に、ランダムフォレスト、SVM（サポートベクターマシン）というクラスも用意されています。詳細は付録 G を参照してください。

```
model.fit(Ⓐ, Ⓑ)
```

Ⓐ: (データ数, 特徴量の数) の形状の 2 次元配列またはデータフレームで表された入力データ

Ⓑ: (データ数,) の形状の 1 次元配列またはシリーズで表されたラベルデータ（出力データ）

ⒶとⒷで指定された入力データとラベルデータからなるデータに対して指定されたモデルのパラメータを学習します。

```
digit_pred = model.predict(Ⓐ)
```

Ⓐ: (データ数, 特徴量の数) の形状の 2 次元配列またはデータフレームで表された入力データ

Ⓐで指定された入力データの各データに対してモデルが予測した値を要素とする (データ数,) の形状の 1 次元配列を返します。

予測した画像の数字のラベルが digit_pred に格納されます。

③以降の演習で手書き数字を予測させる際に、学習～予測の同様の処理を実行するために、関数化しておきます。下記の predict.py を作成して保存してください。

predict.py

```
from sklearn.linear_model import LogisticRegression
from sklearn import datasets
def predictDigits(digit):
    digits = datasets.load_digits()
    X = digits.data
    y = digits.target
    model = LogisticRegression(solver='liblinear')
    model.fit(X, y)
    digit_pred = model.predict(digit)
    return digit_pred
```

Mac 環境のみ先頭行に以下のコードを追加してください。

```
#!/usr/bin/env python3
```

演習⑥ 手書き数字認識システムの構築

Studuino:bit で手書きの数字を撮影し、同一 LAN 内にある Web サーバへ送信します。

Web サーバ側では機械学習ライブラリ scikit-learn で受信した画像から数字を予測し、その数値データを Studuino:bit に返します。

Studuino:bit は予測された数字を LED に表示します。

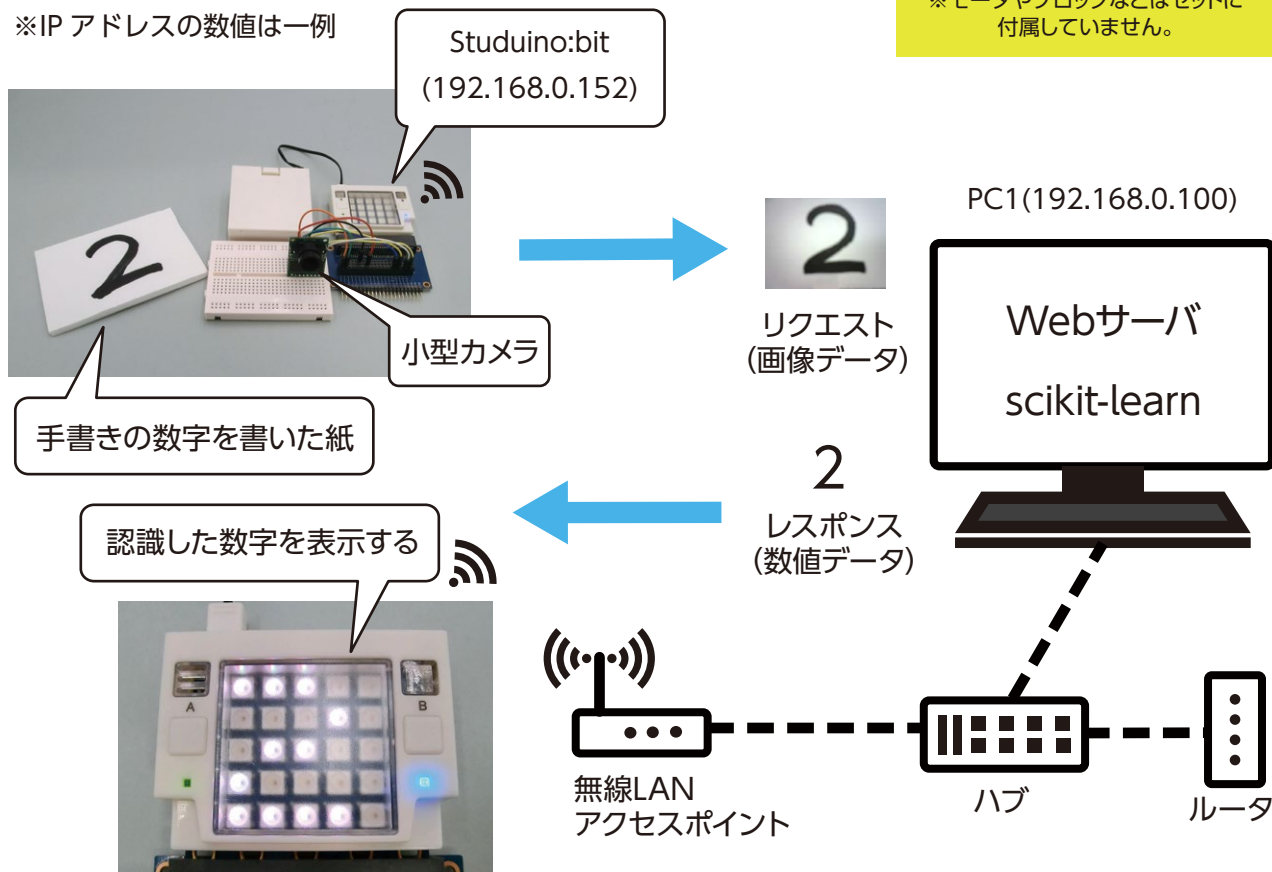
作例動画はコチラ



<https://youtu.be/lyB6IHxjzlo>

※モータやブロックなどはセットに付属していません。

※IP アドレスの数値は一例



無線LAN アクセスポイントとしてWi-Fiルータを使用する場合は、Wi-Fiルータの取扱説明書を参照のうえ、ルータモード(RTモード)をOFFにして、アクセスポイントモード(APモード)に設定してください。

無線LAN アクセスポイントとしてモバイルホットスポットを利用する場合は、付録Eを参照してください。

モバイルホットスポットとは、有線LAN で接続されたWindows10 やWindows11 のコンピュータをWi-Fiルータ化する機能です。なお、図中のルータやハブ、PC1は、コンピュータ教室に設置されている既存の装置を使用します。

ハブの空きポートに無線LANアクセスポイントをLANケーブルで接続し、これにStuduino:bitを無線で接続します。

1. Studuino:bit 側のプログラム作成

①演習②で小型カメラを使用するために必要なライブラリが転送済みの Studuino:bit、小型カメラを接続済みのブレッドボードを使用します。Studuino:bit と PC を接続します。

②下記の LR.py を Mu エディタで作成し、Studuino:bit に転送します。

LR.py は学習テーマ 2 演習②で作成した cap.py に青文字を追記したものです。

LR.py

```
from pystubit.board import display,button_a,button_b
from pystubit_iot import wifi_ifconfig, wifi_config, wifi_connect, wifi_disconnect
import capture
import transfer
import time
url = 'http://Web サーバの IP アドレス : ポート番号 /cgi-bin/device2.py'
fname = 'image.jpg'
wifi_isconnected = 0
wifi_config(ssid=' アクセスポイントの SSID', pwd=' パスワード')
while True:
    if button_b.is_pressed():
        if wifi_isconnected == 0:
            res = wifi_connect(trytime=5)
            wifi_ifconfig(('A', 'B', 'C', 'D'))
            wifi_isconnected = 1
            time.sleep(1)
        elif wifi_isconnected == 1:
            wifi_disconnect()
            wifi_isconnected = 0
            time.sleep(1)
    if button_a.is_pressed():
        time.sleep(1)
        display.clear()
        display.set_pixel(2,2,(10,0,0))
        capture.run(fname)
        display.set_pixel(0,2,(0,10,0))
        display.set_pixel(2,2,(0,10,0))
        display.set_pixel(4,2,(0,10,0))
        ans = 'false'
        while ans == 'false':
            ans = transfer.run(url, fname)
        print('ANS:',ans)
        display.show(str(ans),color=(5,5,5))
        time.sleep(3)
        display.clear()
```

2. サーバ側のプログラム作成

①下記の device2.py を IDLE で作成します。

device2.py は Studuino:bit から手書き数字の画像データを受信して、画像処理を行うプログラム image.py に画像データを渡します。

その後、数値化された手書き数字のデータを予測を行うプログラム predict.py に渡し、返ってきた予測結果を Studuino:bit に送信します。

※Python3.13 以降のバージョンで実行する場合は cgi モジュールが使用できないため、付録 H を参照してください。

Windows の場合

device2.py

```
import cgi
import predict
import image
# ファイルにバイナリモードでデータを書き込む関数
def fwriteb(data,fn):
    file = open(fn, mode='wb')
    file.write(data)
    file.close()

print('Content-Type: text/plain\n')

# Studuino:bit から送られてきた画像を temp.jpg に保存
ftemp= 'cgi-bin/temp.jpg'
param = cgi.FieldStorage()
val = param.getvalue('img') # キー img から値 val (画像データ) を取り出す
fwriteb(val, ftemp)

# 画像を 1 行 64 列の二次元配列 digit に変換
digit = image.imageToData(ftemp)
# digit からラベルの数字を予測する
digit_pred = predict.predictDigits(digit)
print(digit_pred[0])
```

```
#!/usr/bin/env python3
import cgi
import predict
import image
# ファイルにバイナリモードでデータを書き込む関数
def fwriteb(data,fn):
    file = open(fn, mode='wb')
    file.write(data)
    file.close()

print('Content-Type: text/plain\n', end='\n\n')

# Studuino:bit から送られてきた画像を temp.jpg に保存
ftemp= 'cgi-bin/temp.jpg'
param = cgi.FieldStorage()
val = param.getvalue('img') # キー img から値 val (画像データ) を取り出す
fwriteb(val, ftemp)

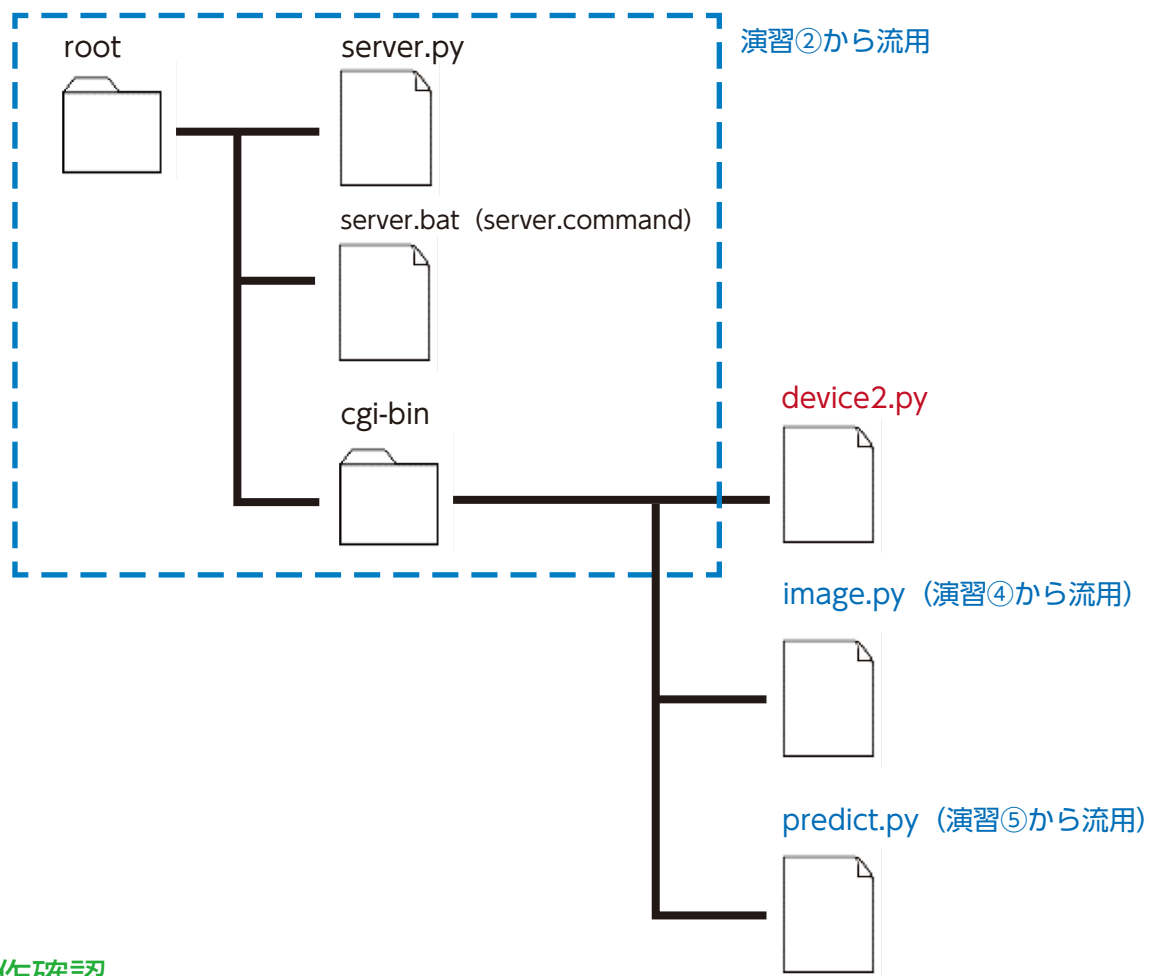
# 画像を 1 行 64 列の二次元配列 digit に変換
digit = image.imageToData(ftemp)
# digit からラベルの数字を予測する
digit_pred = predict.predictDigits(digit)
print(digit_pred[0], end='\n\n')
```

モジュール内の関数を呼び出すには、モジュールを import し、モジュール名 . 関数名 (引数) とします。
device2.py では、p.71 の image.py をモジュールとして import し、image.imageToData(ftemp) によって image モジュール内の imageToData 関数を呼び出しています。
また、p.74 の predict.py をモジュールとして import し、predict.predictDigits(digit) によって predict モジュール内の predictDigits 関数を呼び出しています。

●生成される `__pycache__` フォルダについて

Python ではモジュールの読み込みを高速化するため、自作したモジュールを import し、実行する時にコンパイルしたモジュールが `__pycache__` フォルダの中の `pyc` ファイルにキャッシュ（一時的な履歴データ）が保存されます。キャッシュを残しておけばコンパイルを経ずにプログラムを実行することができるので実行が速くなります。pyc ファイルは、プログラムを変更するたびに書き換えられます。削除しても問題ありませんが、実行が少し遅くなります。

②演習②のディレクトリを使用します。device2_1.pyを削除し、①で作成したdevice2.pyと、cgi-binフォルダ下に演習④で作成したimage.pyと、演習⑤で作成したpredict.pyを格納します。



3. 動作確認

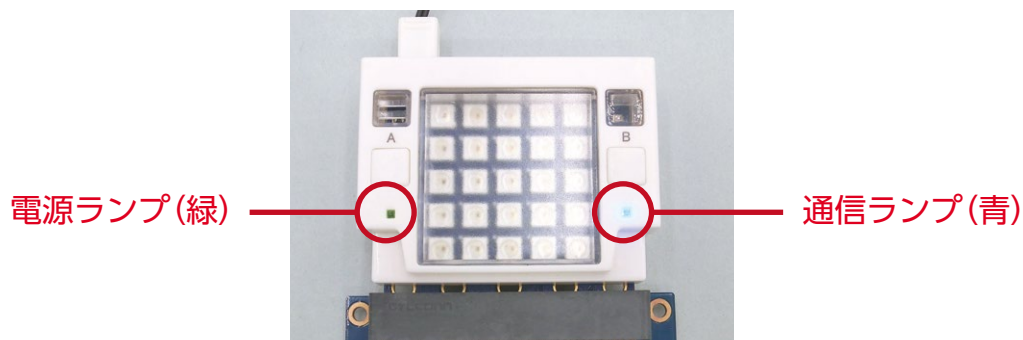
①server.bat (Mac の場合は server.command) をダブルクリックして、Webサーバを起動します。

②Studuino:bitをPCから取り外し、電池ボックスまたはUSBケーブルで給電します。Studuino:bitのリセットボタンを押してLR.pyを実行します。

電源ランプ (緑) が点灯していることを確認し、Bボタンを押して、Studuino:bitをWi-Fiに接続します。

Wi-Fiに接続されると通信ランプ (青) が点灯します。

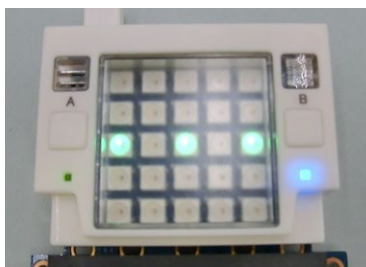
※以降の手順において、動作しない場合はStuduino:bitのリセットボタンを押して再起動を行ってください。



- ③Aボタンを押すと、Studuino:bitのLEDディスプレイに1つの赤色の点が表示された後にカメラが起動して手書き数字が撮影されます。



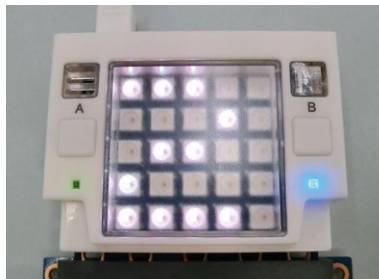
- ④LEDディスプレイに3つの緑色の点が表示され、Webサーバに画像データが送られます。



- ⑤Webサーバ内で認識されたあと、Studuino:bitに認識した数字データが送られ、LED ディスプレイに数字が表示されます。

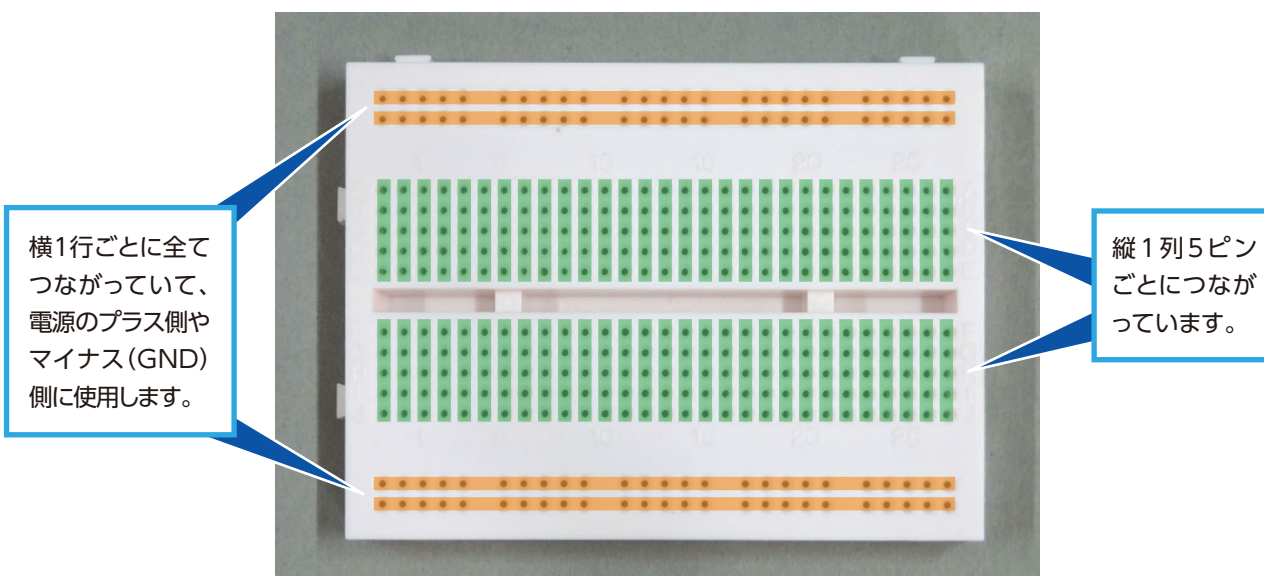
続けてAボタンを押すとカメラが起動して次の手書き数字が撮影されます。

途中でBボタンを押すとWi-Fi接続が切れます。再度接続したい場合はBボタンを押してください。



付録 A. ブレッドボードの使い方

ブレッドボードは内部がつながっており、部品を差し込むだけで電子回路を組み立てることができます。

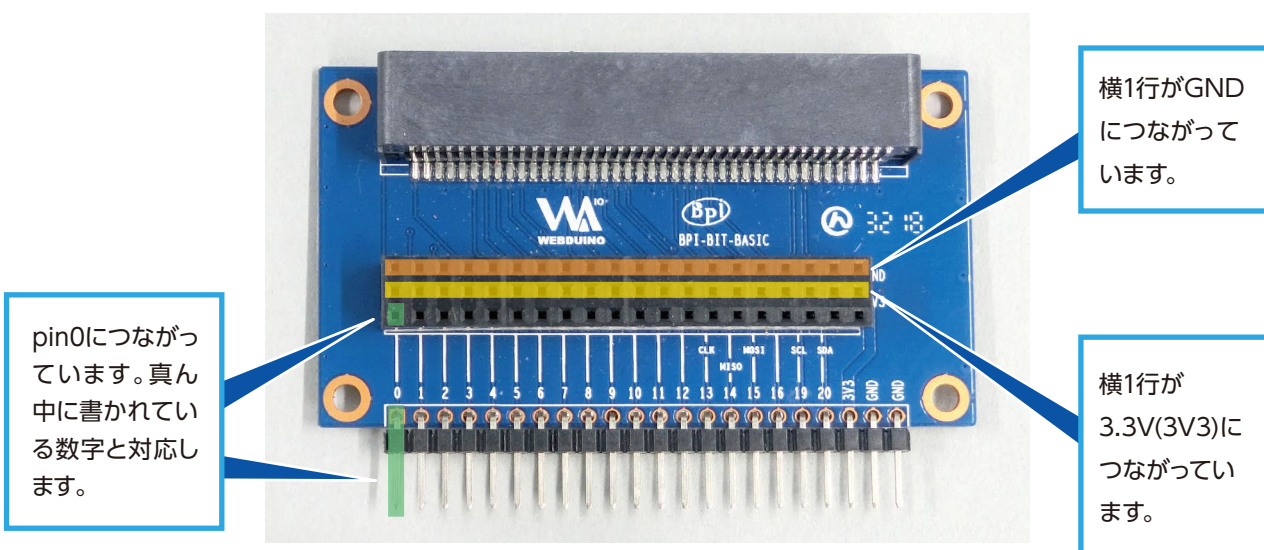


横1行ごとに全てつながっていて、電源のプラス側やマイナス (GND) 側に使用します。

縦1列5ピンごとに繋がっています。

付録 B. 変換エッジコネクタの使い方

Studuino:bit下部のエッジコネクタ主要端子全てにアクセスすることを可能にする、変換基板です。ジャンパワイヤを介して、回路の実装などが簡単に行えます。



pin0につながっています。真ん中に書かれている数字と対応します。

横1行がGNDにつながっています。

横1行が3.3V(3V3)につながっています。

注意事項

エッジコネクタは金属がむき出しになっている部分があります。そこに金属製のものが接触してショートしないように注意してください。

付録C. displayオブジェクトの機能一覧

メソッド	説明
<code>get_pixel(x, y)</code>	x 列 y 行の LED の色を返します。返り値は (R,G,B) です。
<code>set_pixel(x, y, color)</code>	x 列 y 行の LED の色を color で設定します。color は、(R,G,B), [R,G,B], #RGB で指定します。
<code>clear()</code>	すべての LED の明るさを 0 (オフ) に設定します。
<code>show(iterable, delay=400, *, wait=True, loop=False, clear=False, color=None)</code>	iterable(イメージ / 文字列 / 数字) を順番に表示します。
<code>scroll(string, delay=150, *, wait=True, loop=False, color=None)</code>	string(文字 / 数字) を水平方向にスクロールさせます。
<code>on()</code>	ディスプレイの電源を ON にします。
<code>off()</code>	ディスプレイの電源を OFF にします。(ディスプレイに関連づけられた GPIO 端子を他の目的に再利用できるようにします)。
<code>is_on()</code>	ディスプレイの電源が ON であれば「True」を、OFF であれば「False」を返します。

LEDの色を設定する下記のプロパティを持ちます。

BLACK (消灯)、WHITE (白)、RED (赤)、LIME (ライム)、BLUE (青)、YELLOW (黄)、CYAN (シアン)、MAGENTA (マゼンタ)、SILVER (銀)、GRAY (灰)、MAROON (茶色)、OLIVE (オリーブ)、GREEN (緑)、PURPLE (紫)、TEAL (青緑)、NAVY (紺)

付録 D. Wi-Fi 機能の一覧

メソッド	説明
wifi_config(ssid='Wi-Fi ルータの SSID', pwd=' Wi-Fi ルータのパスワード')	Studuino:bit を接続する Wi-Fi の設定を引数に設定します。
wifi_ifconfig(('Ⓐ','Ⓑ','Ⓒ','Ⓓ'))	Studuino:bit に固定の IP アドレスを設定します。 Ⓐ Studuino:bit に割り振る IP アドレス Ⓑ サブネットマスク Ⓒ デフォルトゲートウェイ Ⓓ DNS サーバ
wifi_connect(Ⓐ)	Studuino:bit を Wi-Fi 接続します。 Ⓐ: trytime= 再接続回数
wifi_disconnect()	Studuino:bit の Wi-Fi 接続を解除します。

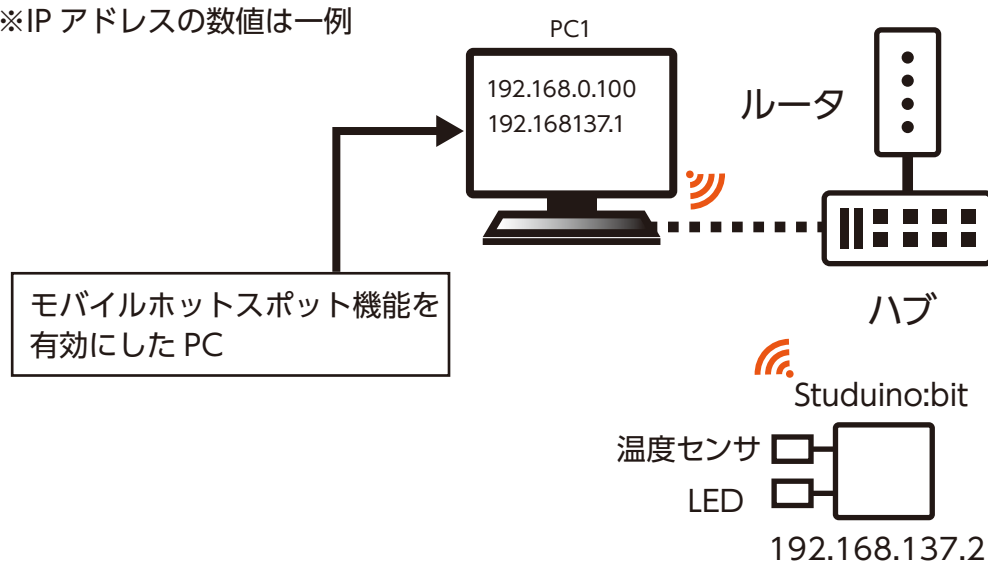
付録 E. モバイルホットスポットの使い方

モバイルホットスポットとはインターネットに接続されたWindows 10 やWindows11のPCを無線LANアクセスポイントに変換する機能です。

図中のルータやハブ、PC1は、コンピュータ教室などに設置されている既存の装置を使用します。PC1にモバイルホットスポット機能がない場合は、モバイルホットスポット機能があるPCをハブに接続し、そのPCにStuduino:bitを無線LAN接続させることもできます。

なお、モバイルホットスポット機能を有効にしたPC（図ではPC1）には、有線LAN側のIPアドレス（図では192.168.0.100）とは別に、モバイルホットスポット側に192.168.137.1がゲートウェイとして割り当てられるため、Studuino:bitに割り当てるIPアドレスのネットワーク部は、192.168.137にする必要があります。図では、Studuino:bitのIPアドレスとして192.168.137.2を割り当てています。

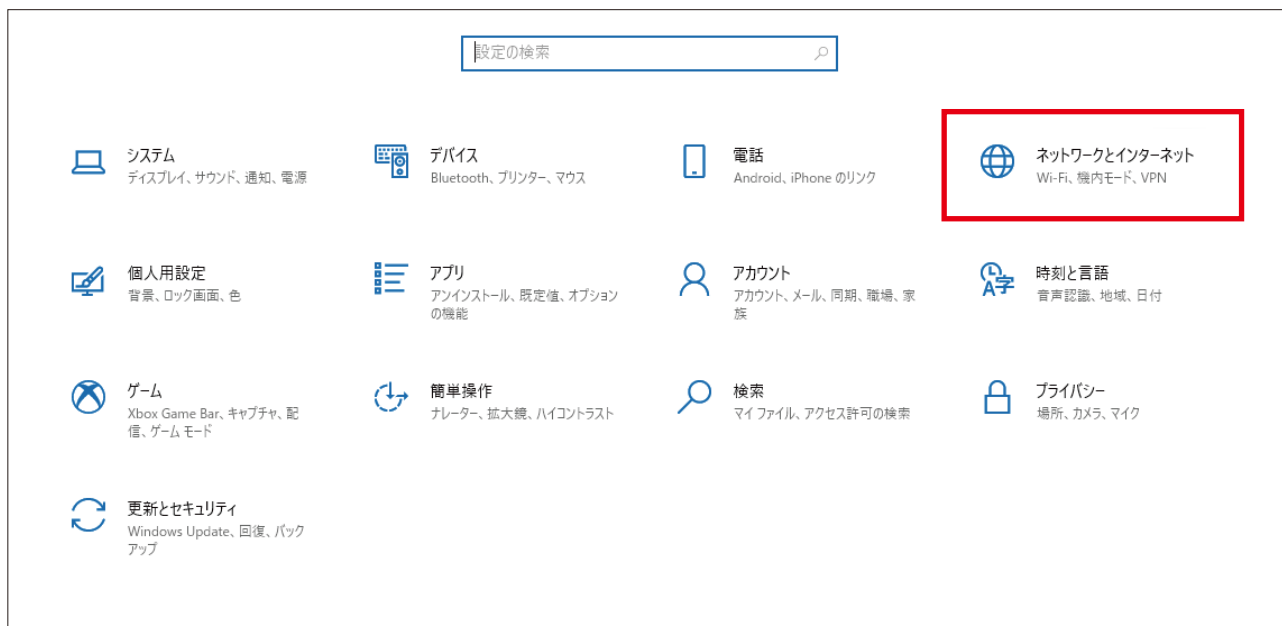
※IP アドレスの数値は一例



- ①「スタートボタン」→歯車の形の「設定」をクリックするか、キーボードの「[Windows] キー+「I」 キーを押して、設定画面を表示します。



②「ネットワークとインターネット」をクリックします。



③「モバイルホットスポット」をクリックします。



- ④複数のインターネット接続がある場合は、共有するインターネット接続を選択して、「Wi-Fi」を選択します。

モバイル ホットスポット

インターネット接続を他のデバイスと共有します

☐ オフ

インターネット接続を共有する:

ネットワーク 7

インターネット接続を共有する

☒ Wi-Fi

☐ Bluetooth

ネットワーク名: XXXXXXXXXXXXXXX

ネットワーク パスワード: XXXXXXXXXXXXXXX

編集

複数ある場合は選択する

- ⑤モバイルホットスポットを「オン」にします。ネットワーク名とネットワークパスワードの部分に表示される値を、Studuino:bit の wifi_config に設定して、モバイルホットスポットに接続します。

モバイル ホットスポット

インターネット接続を他のデバイスと共有します

☒ オン

インターネット接続を共有する:

ネットワーク 7

インターネット接続を共有する

☒ Wi-Fi

☐ Bluetooth

ネットワーク名: XXXXXXXXXXXXXXX

ネットワーク パスワード: XXXXXXXXXXXXXXX

編集

接続されているデバイス: 0 / 8

省電力

デバイスが接続されていないときに、モバイル ホットスポットを自動的にオフにします。

☐ オフ

wifi_config(ssid=' ネットワーク名',
pwd=' ネットワークパスワード')

●モバイルホットスポットの設定画面に表示される IP アドレスについて

モバイルホットスポットの設定画面には接続されているデバイスの「デバイス名」「IP アドレス」「物理アドレス (MAC)」が表示されます。

Studuino:bit 側で `wifi_ifconfig` 関数を使用して「192.168.137.2」を割り当てている場合も、モバイルホットスポットの設定画面には別の IP アドレス（下記の画像の例では「192.168.137.138」）が表示されてしまいます。

これは初めに Studuino:bit がモバイルホットスポットに接続したときに「192.168.137.138」が DHCP 機能によって動的に割り当てられ、その後指定した固定 IP アドレスの「192.168.137.2」が割り当て直されることによって発生します。

```
from pystubit_iot import wifi_ifconfig, wifi_config, wifi_connect
wifi_config(ssid=' ネットワーク名 ',pwd=' ネットワークパスワード ')
wifi_ifconfig(('192.168.137.2', '255.255.255.0', '192.168.137.1', '192.168.137.1'))
wifi_connect()
```

最終的にはこの設定が割り当てられている

Studuino:bit 側のプログラム



接続されているデバイス: 1 / 8

デバイス名	IP アドレス	物理アドレス (MAC)
espressif	192.168.137.138	30:ae:a4:cd:2f:9c

`wifi_config` 関数を使わない DHCP 機能による動的 IP アドレスを割り当てた場合は、モバイルホットスポットの設定画面に表示された IP アドレスが、Studuino:bit の IP アドレスです。

付録 F. scikit-learn の手書き数字のデータセット

下記の imageF.py を実行すると scikit-learn の手書き数字のデータセットを数字ごとに全て確認できます。

赤字部分の 0 を 0 ～ 9 に打ち換えて実行してください。

imageF.py

```
from sklearn import datasets
import matplotlib.pyplot as plt

digits = datasets.load_digits()
j = 0

for i in range(1797):
    if digits.target[i] == 0:
        plt.subplot(14, 14, j+1)
        plt.axis('off')
        plt.imshow(digits.images[i], cmap='Greys')
        j += 1
plt.show()
```

プログラムを実行すると下記のように表示されます。

7 には少々見慣れない書き方 (横線が入った書き方) が多く含まれています。scikit-learn のデータセットを使って機械学習させた場合、7 はデータセットと同じような書き方をすると正しく認識されやすくなります。

付録 G. 機械学習モデルの比較

演習⑤、演習⑥では機械学習のアルゴリズムとしてLogisticRegression (ロジスティック回帰) を使用しました。scikit-learnにはランダムフォレストや、SVM(サポートベクターマシン) というアルゴリズムも用意されており、アルゴリズムが異なると予測精度(結果)も異なります。

アルゴリズムを変更する場合、predict.pyを下記のように書き換えてください。

SVM(サポートベクターマシン) の記述例

predict.py

```
from sklearn.linear_model import LogisticRegression
from sklearn import datasets
def predictDigits(digit):
    digits = datasets.load_digits()
    X = digits.data
    y = digits.target
    model = sklearn.svm.SVC(gamma=0.001)
    model.fit(X, y)
    digit_pred = model.predict(digit)
    return digit_pred
```

Mac 環境のみ先頭行に以下のコードを追加してください。

```
#!/usr/bin/env python3
```

ランダムフォレストの記述例

predict.py

```
from sklearn.linear_model import LogisticRegression
from sklearn import datasets
def predictDigits(digit):
    digits = datasets.load_digits()
    X = digits.data
    y = digits.target
    model = RandomForestClassifier(random_state=0, n_estimators=10)
    model.fit(X, y)
    digit_pred = model.predict(digit)
    return digit_pred
```

Mac 環境のみ先頭行に以下のコードを追加してください。

```
#!/usr/bin/env python3
```

付録 H. cgi モジュールを使用しないプログラム

演習②、演習⑥ではクライアントから送信されたデータをcgiモジュールで取得しています。

Python3.11.0でcgiモジュールを実行すると「DeprecationWarning: 'cgi' is deprecated and slated for removal in Python 3.13 (cgiモジュールはPython3.11から非推奨であり、3.13からは削除される予定です)」という警告メッセージがコマンドプロンプトに表示されます。

今後はcgiモジュールを使ったプログラムをPython3.11やPython3.12で実行すると、警告メッセージは表示されますが、正常に動作すると考えられます。

Python3.13以降のバージョンで実行するとcgiモジュールが削除されるため、動作しないことが予想されます。

演習②、演習⑥のプログラムにおいて、Windows環境でcgiモジュールを使わずにクライアントから送信されたデータをWebサーバで取得するプログラムを本付録に記載します。

Mac 環境では以降に記載する方法には対応していないため、cgi モジュールを使ったプログラムをPython3.12以前のバージョンで実行してください。

演習②

演習②の device2_1.py を下記の通り書き換えます。

※下記の sys モジュールを使用した方法は Python3.12 以前のバージョンでも動作します。

ただし、「値=オブジェクト.getvalue('キー')」のように簡単にキー名を指定して値を取り出すことはできません。

device2_1.py

```
import sys
# ファイルにバイナリモードでデータを書き込む関数
def fwriteb(data,fn):
    file = open(fn, mode='wb')
    for j in range(4,len(data)-1):
        file.write(data[j])
    file.close()
print('Content-Type: text/plain\n')
ftemp= 'cgi-bin/temp.jpg'
val = sys.stdin.buffer.readlines()
fwriteb(val,ftemp)
```

演習⑥

演習⑥の device2.py を下記の通り書き換えます。

※下記のsysモジュールを使用した方法はPython3.12以前のバージョンでも動作します。ただし、「値=オブジェクト.getvalue('キー')」のように簡単にキー名を指定して値を取り出すことはできません。

device2.py

```
import sys
import predict
import image
# ファイルにバイナリモードでデータを書き込む関数
def fwriteb(data,fn):
    file = open(fn, mode='wb')
    for j in range(4,len(data)-1):
        file.write(data[j])
    file.close()
print('Content-Type: text/plain\n')

# Studuino:bit から送られてきた画像を temp.jpg に保存
print('Content-Type: text/plain¥n')
ftemp= 'cgi-bin/temp.jpg'
val = sys.stdin.buffer.readlines()
fwriteb(val,ftemp)

# 画像を 1 行 64 列の二次元配列 digit に変換
digit = image.imageToData(ftemp)
# digit からラベルの数字を予測する
digit_pred = predict.predictDigits(digit)
print(digit_pred[0])
```

※「¥」は環境によってバックスラッシュ「\」が表示されることがあります。

●画像データに関して

クライアントから送信されるデータには画像データだけでなく、ヘッダやフッタと呼ばれているデータも含まれています。

取得したデータである val(関数内では data) は以下の形式の配列になっています。

[h0, h1, h2, h3, g0, g1, g2, ..., gn, f]

4個の h0 ~ h3 はヘッダであり、f はフッタです。

ヘッダとフッタの間に g0 ~ gn の画像データが入っています。

len(data) は配列の要素数であり、len(data)-1 はフッタ f が入っている配列要素のインデックス番号 (添字) を示すため、g0,g1,g2,...,gn の画像データが入っている配列要素のインデックス番号は 4 から len(data)-1 の1つ手前までになります。

これを cgi-bin フォルダ内の画像ファイル temp.jpg に書き出しています。

カメラ映像によって g0 ~ gn の数が異なるため、len(data) は一定ではありません。