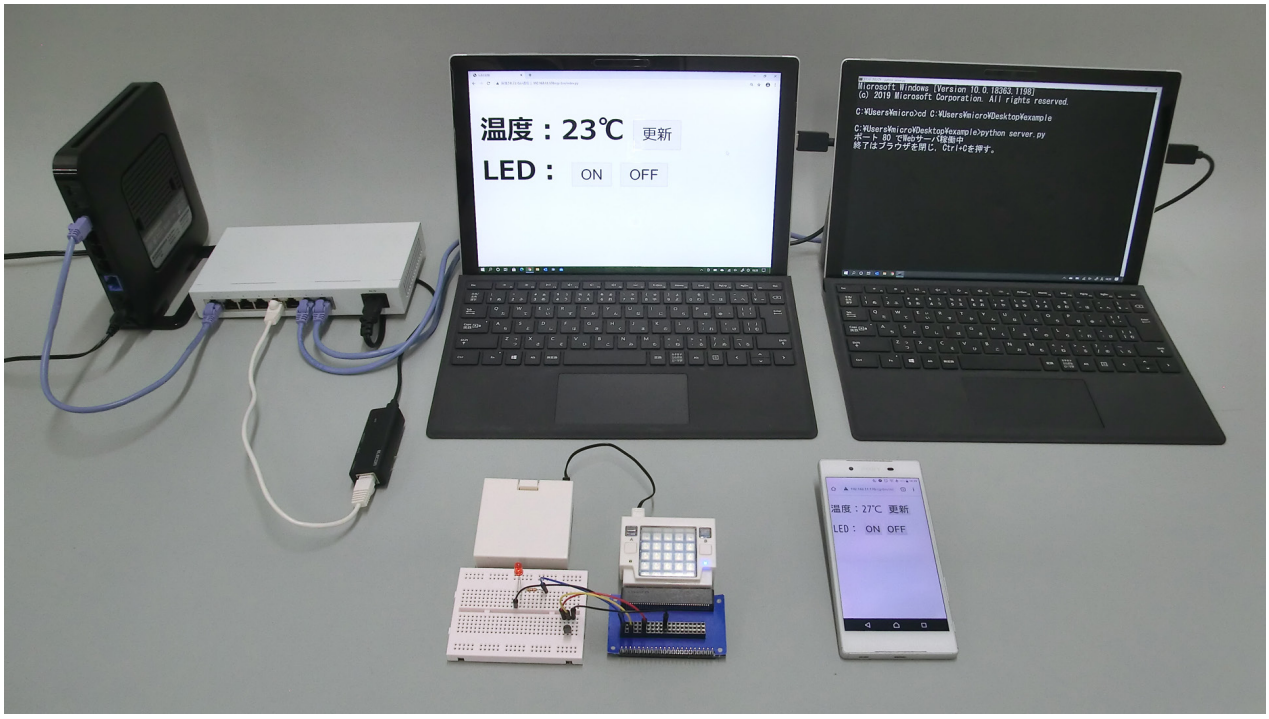
Studuino:bit
スタディーノ ビット

高校情報Ⅱキット
遠隔計測制御システム(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



ArTeC®

目 次

準 備

準備① Pythonの開発環境 …… 3
準備② MicroPython開発環境 …… 10
複数のプログラムを転送した場合の実行方法 …… 15

演 習

演習① 温度センサからのアナログ入力プログラム …… 16
演習② 温度センサからのアナログ入力と LED へのデジタル出力プログラム …… 20
演習③ 温度センサからのアナログ入力と2つのLEDへのデジタル出力プログラム …… 22
演習④ Webサーバの構築 …… 24
Webサーバが起動しない場合のトラブルシューティング …… 32
演習⑤ Webサーバを用いた計測制御システムのモデルの構築 …… 36
wifi_config関数について …… 38
wifi_ifconfig関数について …… 38
演習⑥ 遠隔計測制御システムの構築 …… 46

付 録

付録 A. ブレッドボードの使い方 …… 57
付録 B. 変換エッジコネクタの使い方 …… 57
付録 C. デジタル入出力・アナログ入力オブジェクトの機能一覧 …… 58
付録 D. display オブジェクトの機能一覧 …… 59
付録 E. モバイルホットスポットの使い方 …… 60
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 …… 63
付録 F. Wi-Fi機能の一覧 …… 64
付録 G. cgiモジュールを使用しないプログラム …… 65

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

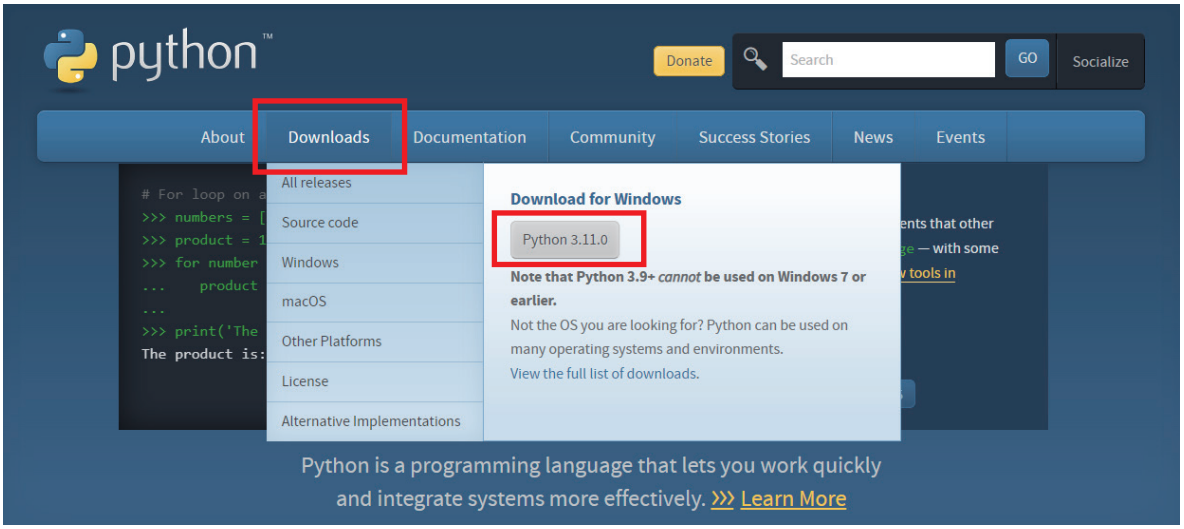
1．Python のインストール

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。

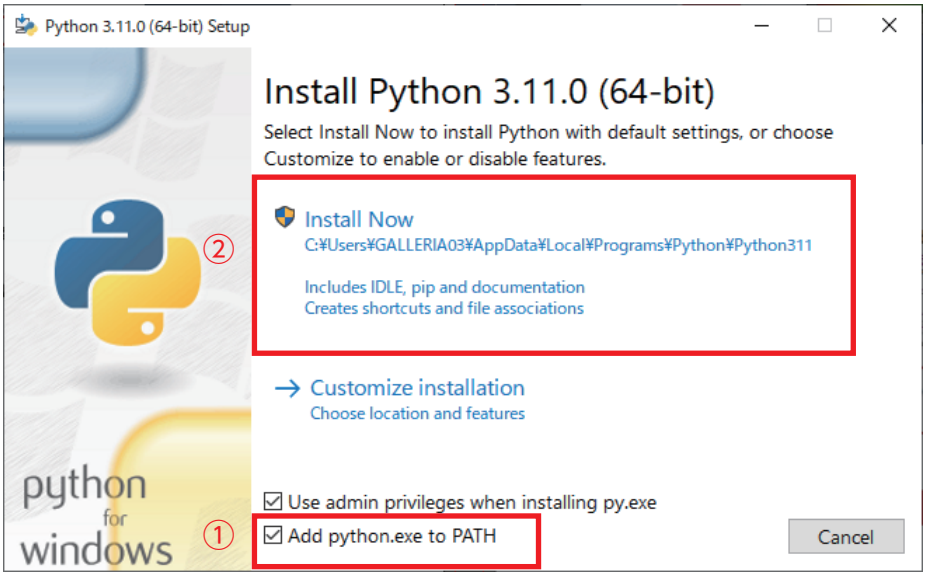
Python3.13以降のバージョンでは正常に動作しない可能性が考えられます。

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。

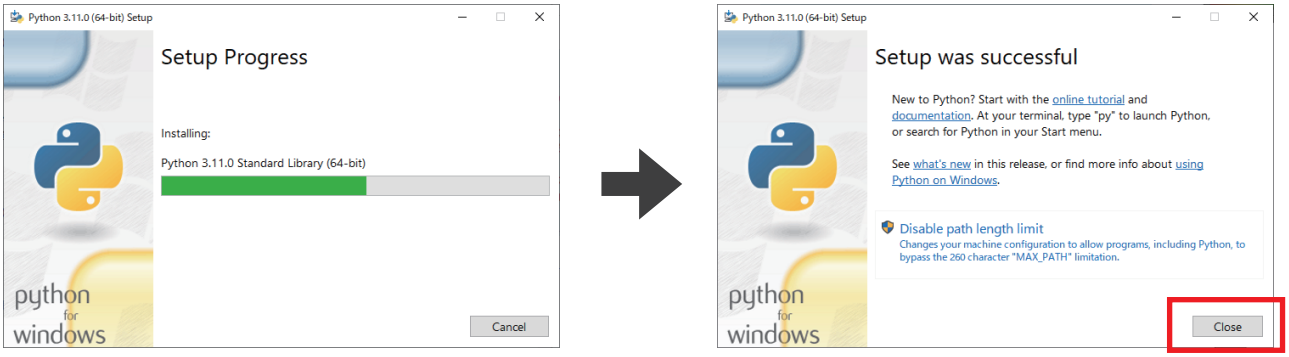
Mac環境ではcgiモジュールを使わない方法には対応していないため、Python3.12以前のバージョンをインストールして、cgiモジュールを使ったプログラムを作成してください。

詳細は付録G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。



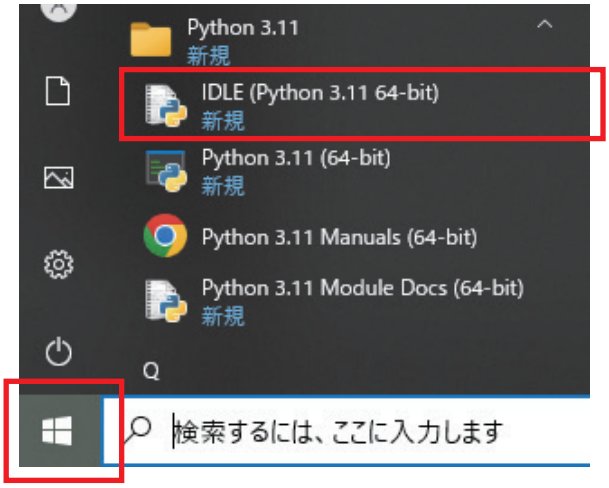
※すでにPythonをインストール済みのPCを使用する場合の注意事項
・PythonのWebサーバ機能を使用する際、②のパスの設定(☑)の有無によって実行方法が異なります。
・本テキストに記載のserver.pyのサンプルプログラムはPython3.7以降のバージョン用に作成されています。Python3.6以前のバージョンでは正常に動作しません。

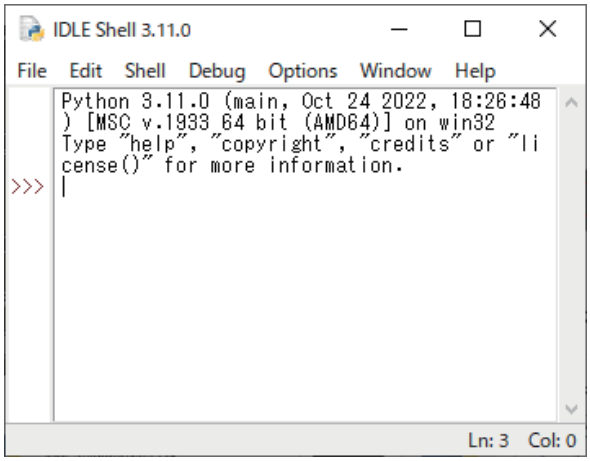
2. IDLE（統合開発環境）の使い方

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。

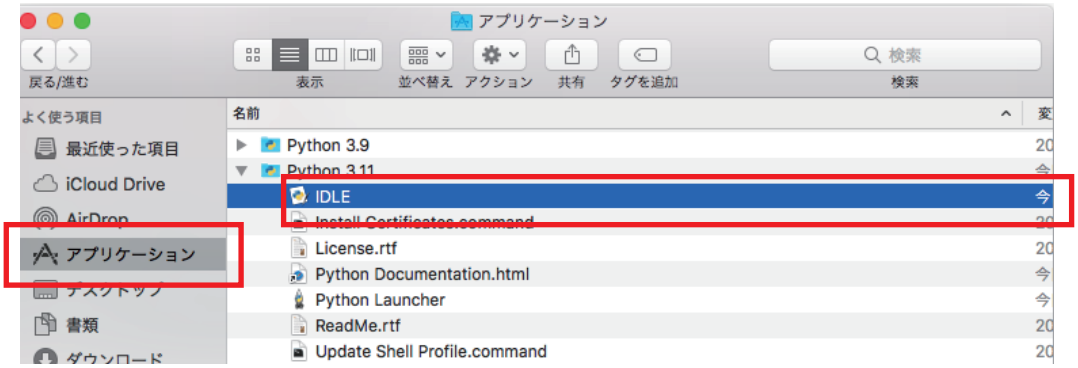




シェルウィンドウ

Mac 環境

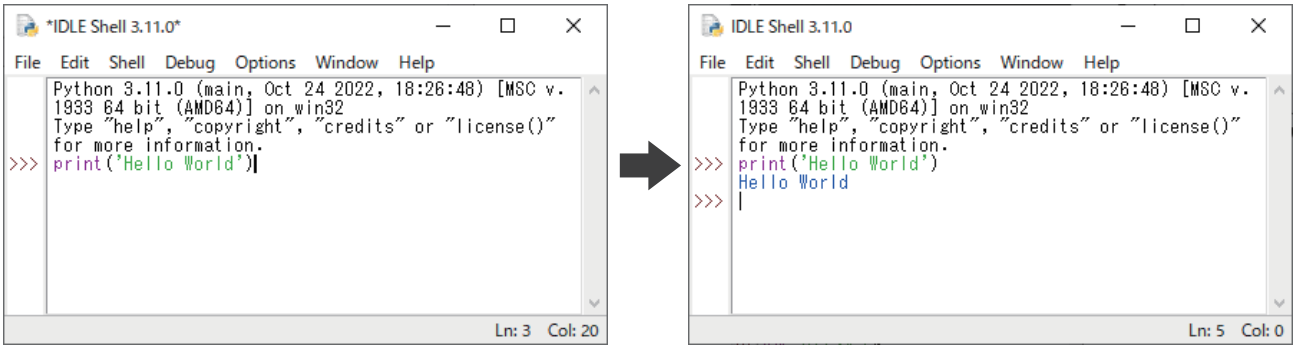
Finderを起動して「アプリケーション」から使用したいバージョンの「Python」フォルダ内にあるIDLEを選択すると、IDLEが起動しシェルウィンドウが表示されます。



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。

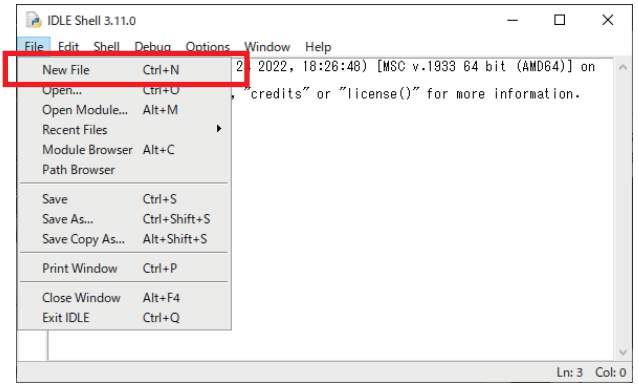


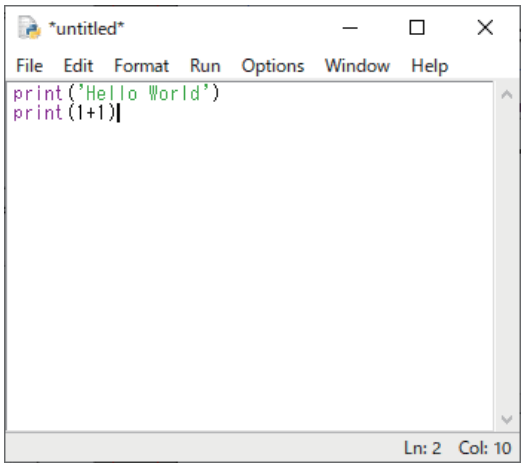
2-2.新規ファイルを作成し、ファイル実行モードで実行する

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。

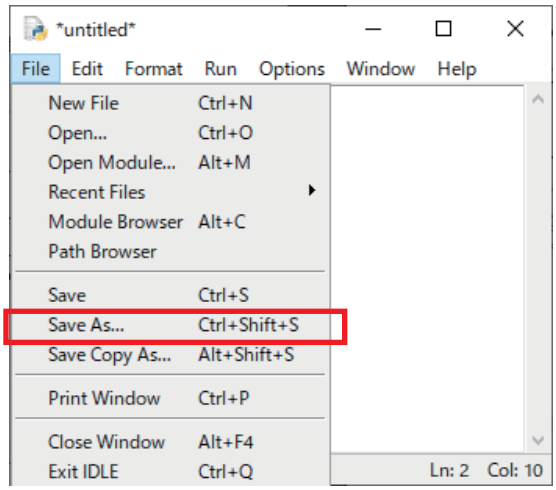
```
print('Hello World')
print(1+1)
```



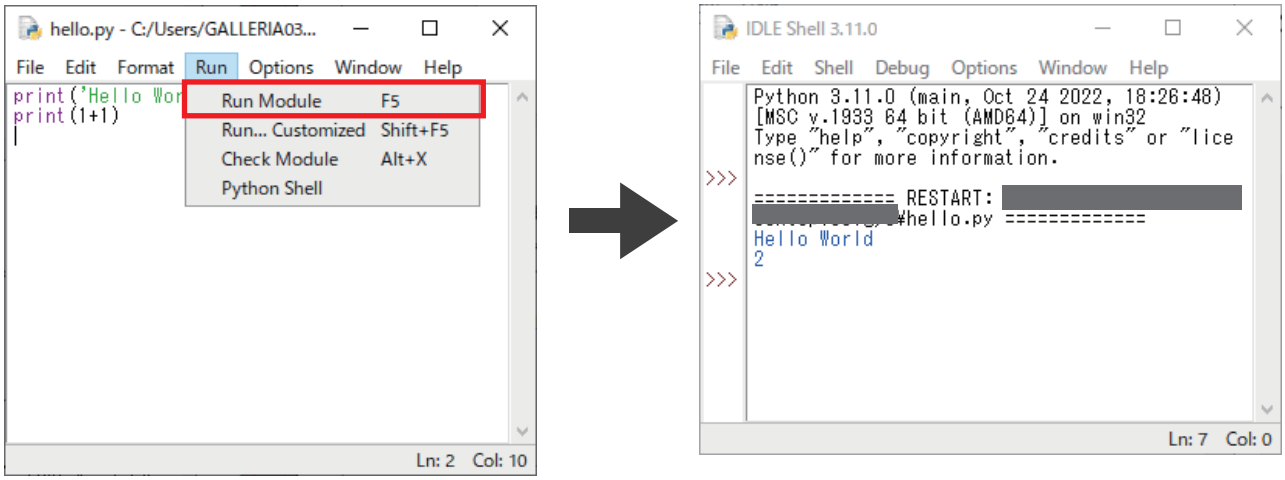


エディタウィンドウ

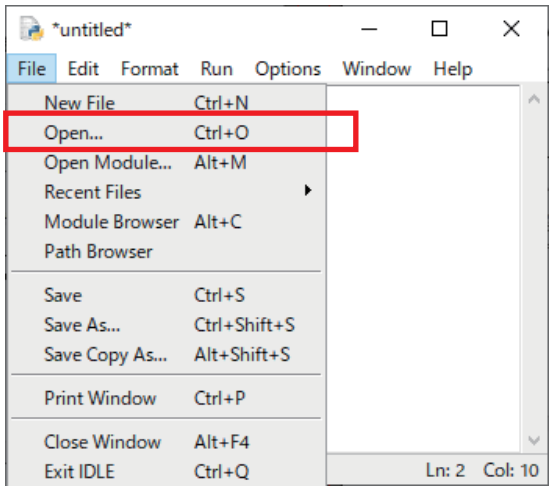
②「File」、「Save As」の順にクリックしてファイルに名前を付けて保存します。



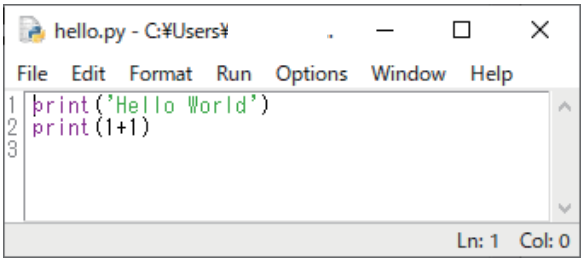
③エディタウィンドウの「Run」、「Run Module」の順にクリックすると、実行結果がシェルウィンドウに表示されます。



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

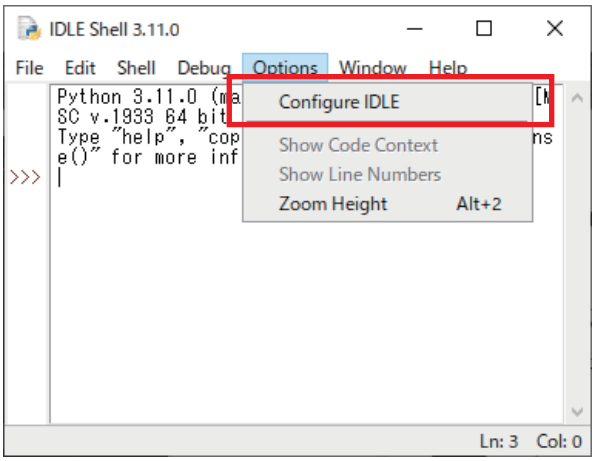


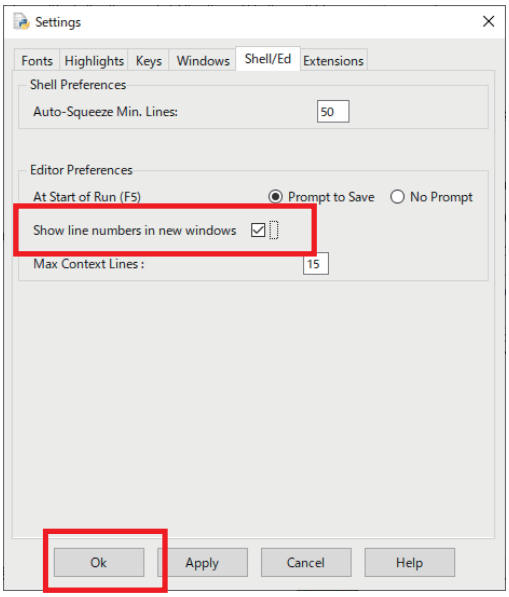
●行番号の表示

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

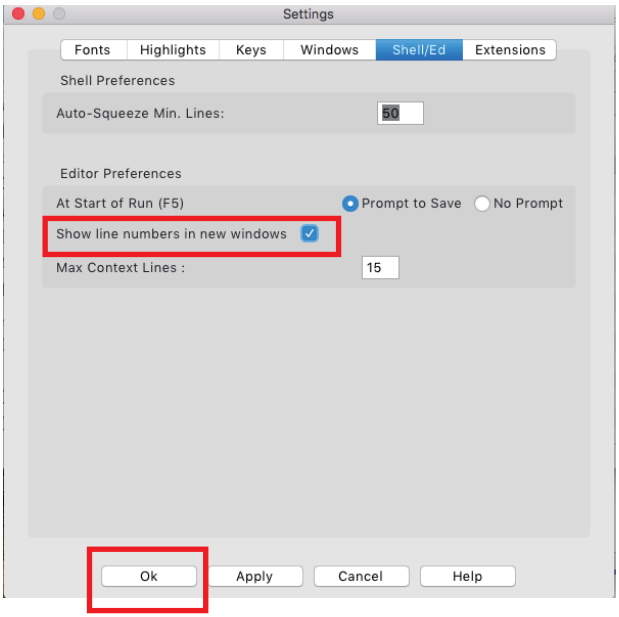
シェルウィンドウまたはエディタウィンドウから「Option」、「Configure IDLE」、「Shell/Ed（またはGeneral）タブ」の順にクリックし、「Show line numbers in new windows」をチェックして「OK」をクリックすると、行番号が表示されます。





Mac 環境

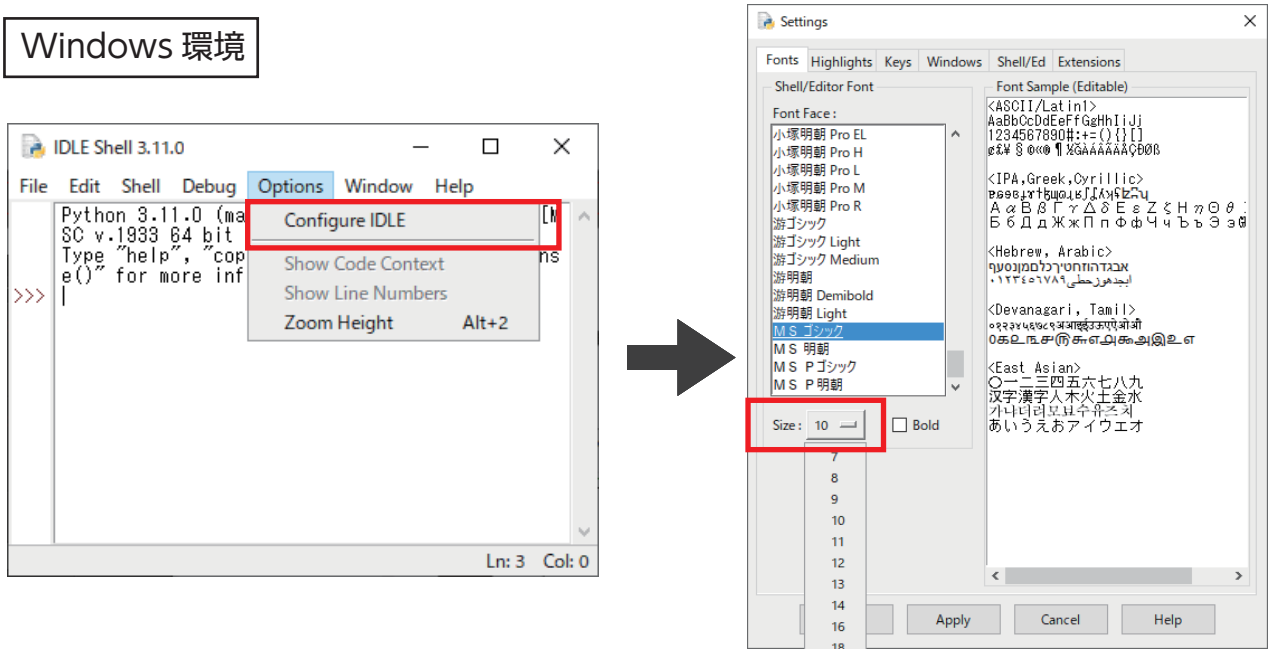
シェルウィンドウまたはエディタウィンドウから「IDLE」、「Preferences」、「Shell/Ed（またはGeneral）タブ」の順にクリックし、「Show line numbers in new windows」をチェックして「OK」をクリックすると、行番号が表示されます。



●文字サイズの変更

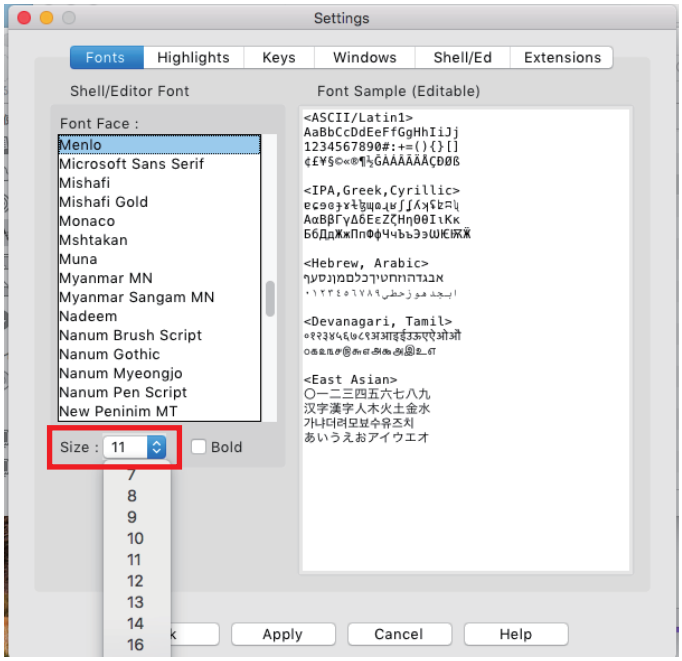
シェルウィンドウまたはエディタウィンドウから「Options」、「Configure IDLE」の順にクリックし、「Size:」横のボタンをクリックして文字サイズを選択できます。「OK」をクリックすると、選択した文字サイズに変更されます。

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

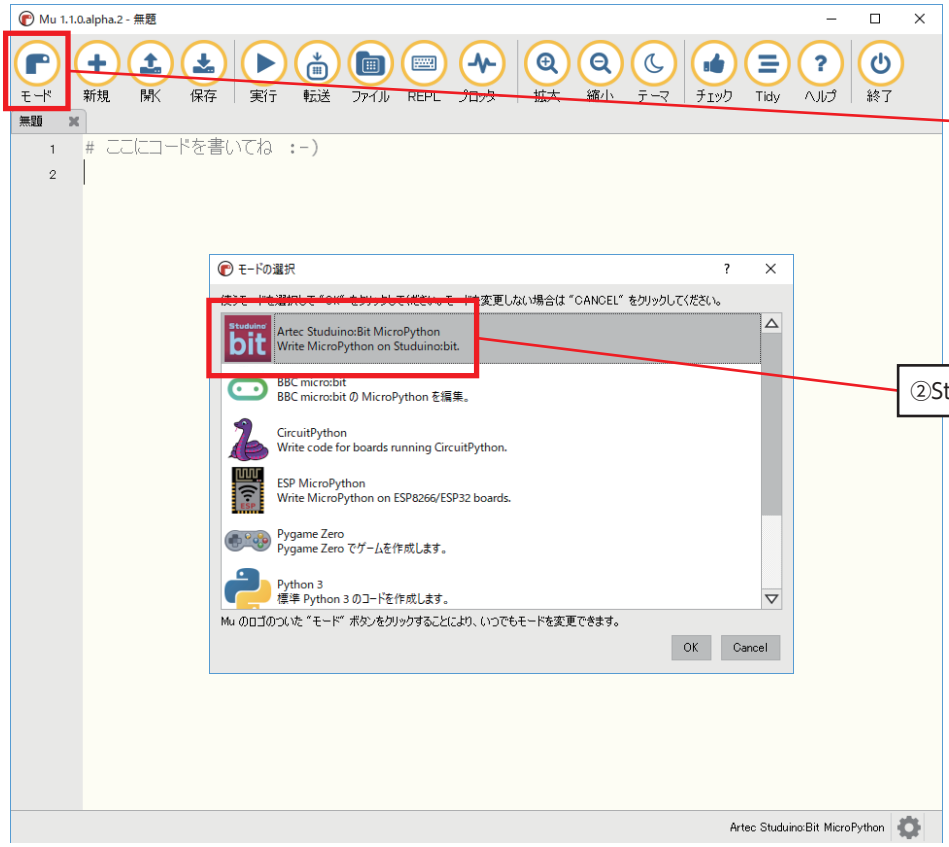
Python	MicroPython
PC 上で動作する Web サーバの役割	Studuino:bit 上で動作する
IDLE でプログラムを作成 	Mu エディタでプログラムを作成 

2. REPL でプログラムを実行する

Python の対話型インタープリタ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。

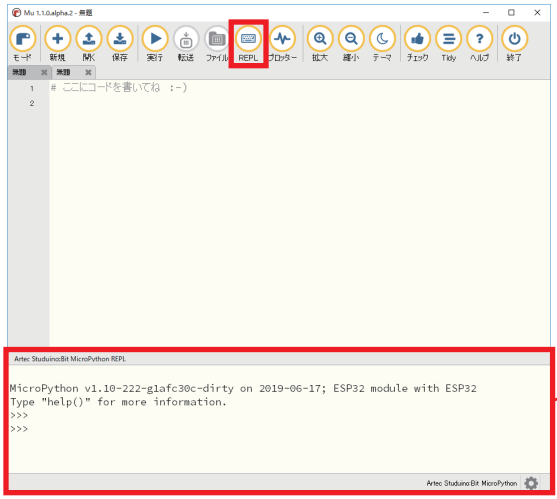
② Muを起動して、下図のように「モード」ボタンをクリックして、「Artec Studuino:bit MicroPython」をダブルクリックしてください。



①モードをクリック

②Studuino:bit をダブルクリック

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。



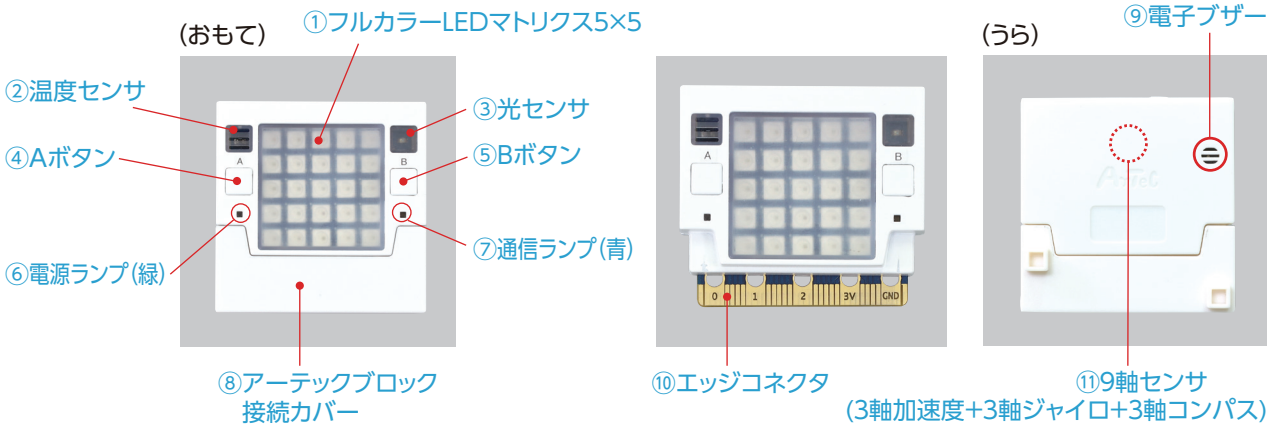
このウィンドウで REPL を使用します。

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。

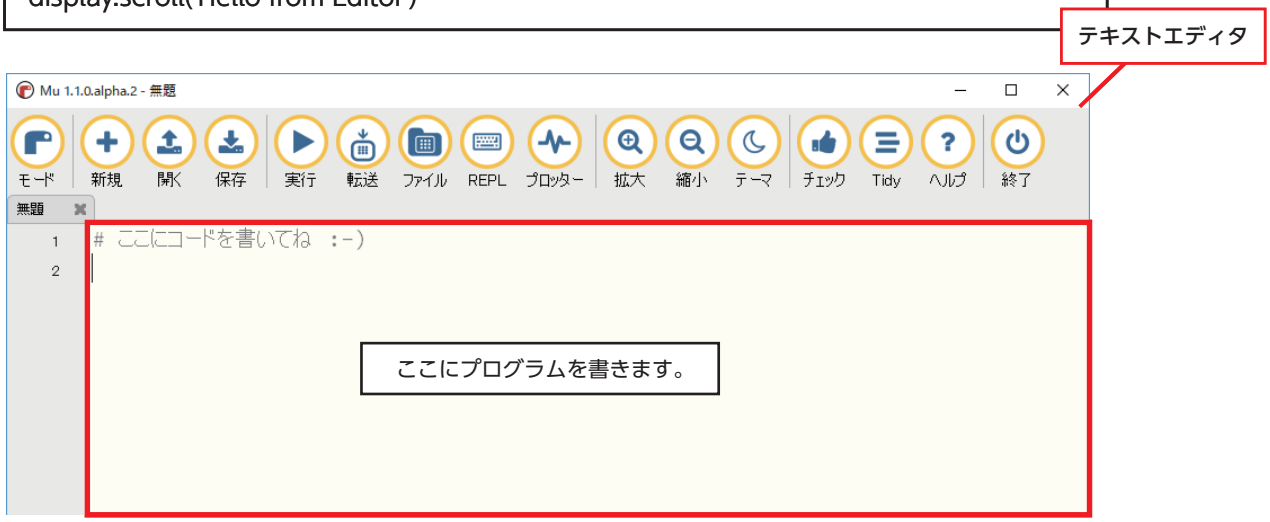
```
>>> from pystubit.board import display
>>> display.scroll('Hello, MicroPython')
```

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。

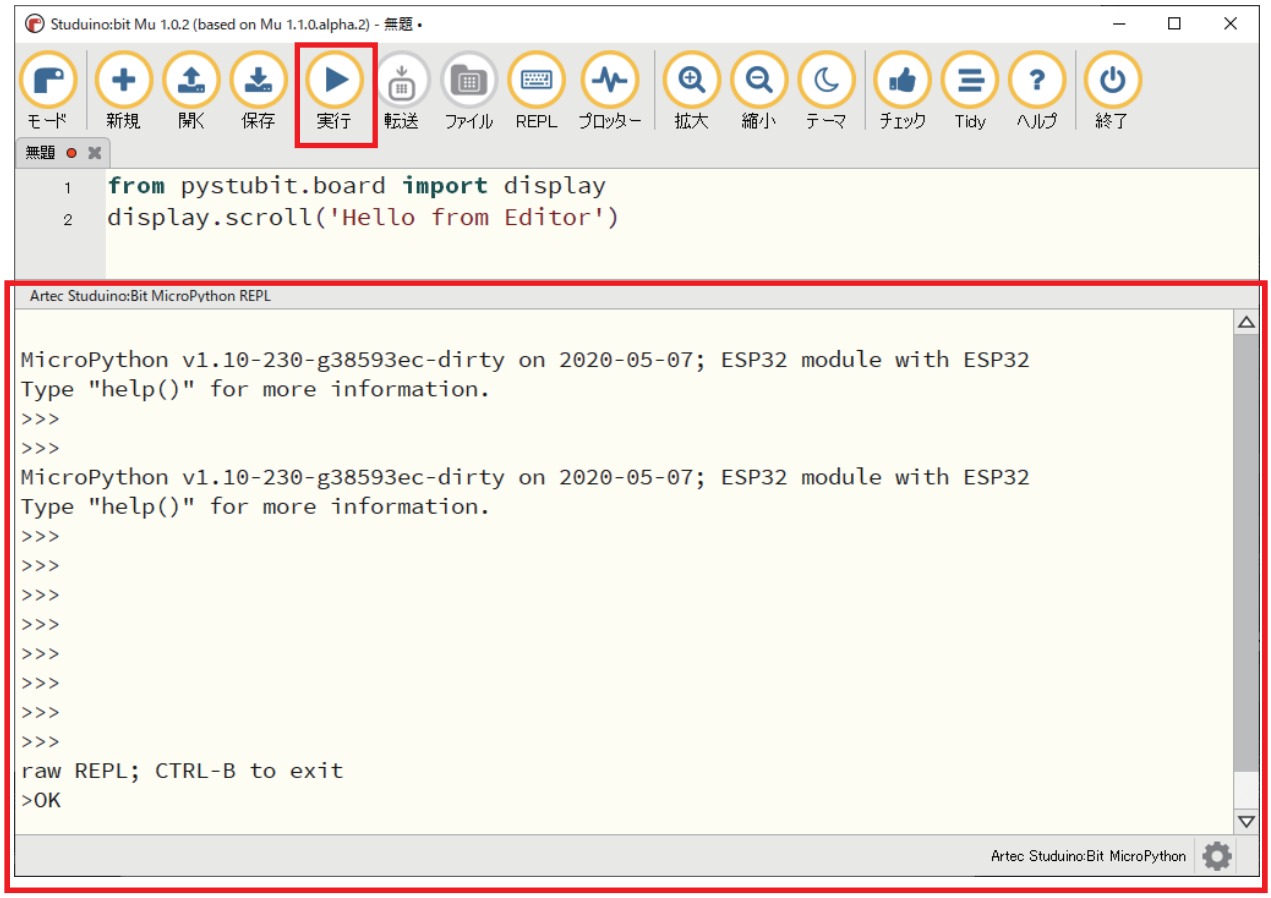
①下記をテキストエディタに入力します。

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



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。



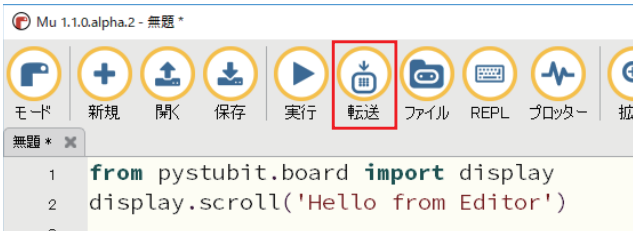
「実行」ボタンをクリックすると下側にウィンドウが表示され、REPLを使用できるようになります。

5.「転送」ボタンでプログラムを実行する

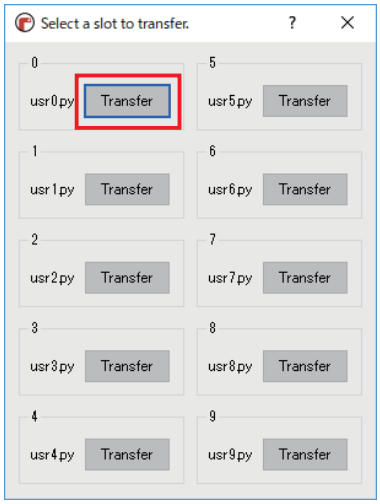
Muエディタで編集したPythonスクリプトをStuduino:bitに転送し、実行する手順を下記に記します。

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。

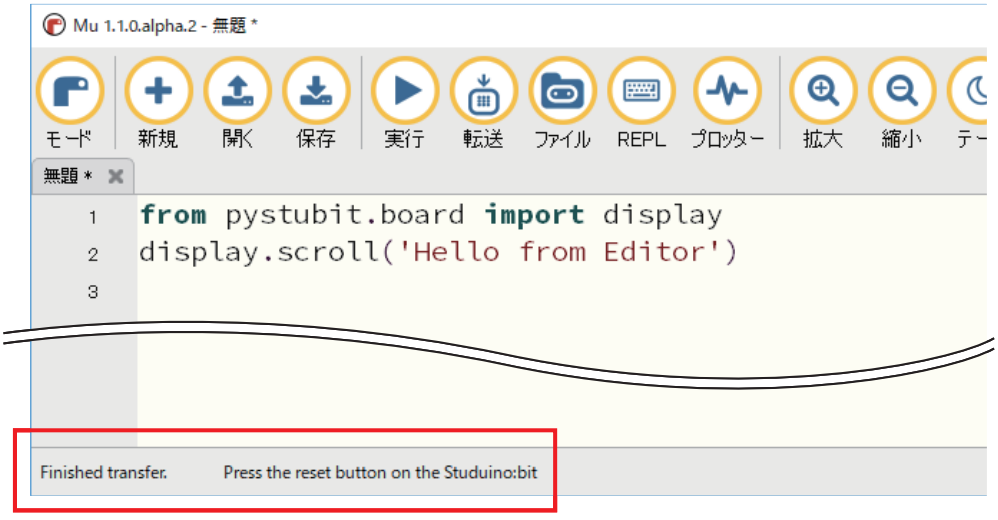
※「転送」ボタンが無効(灰色表示)になっている場合は「REPL」ボタンをクリックし、REPLウィンドウを閉じることで「転送」ボタンが有効になります。



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。

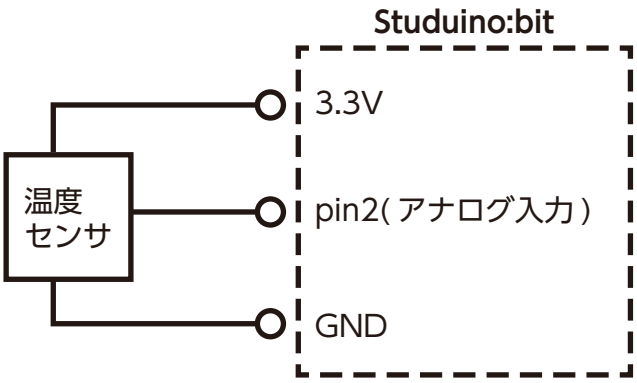


※工場出荷段階で動作確認用に各番号にはサンプルプログラムが書き込まれています。
※サンプルプログラムはソフトウェアを使用して新たなプログラムを書き込むと上書きされます。

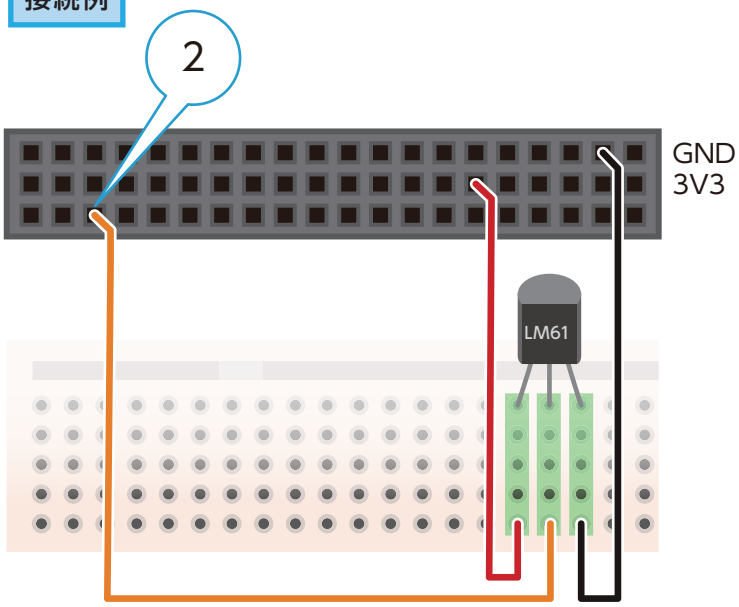
演習① 温度センサからのアナログ入力プログラム

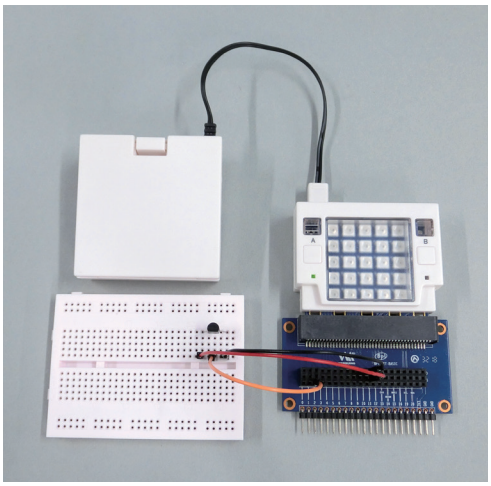
温度センサで周囲の温度を計測し、Studuino:bitに温度を表示するプログラムを作成します。
Studuino:bit本体内部には温度センサが内蔵されていますが、ここでは外部に接続した温度センサを使って計測する方法について説明します。

①温度センサとStuduino:bit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。

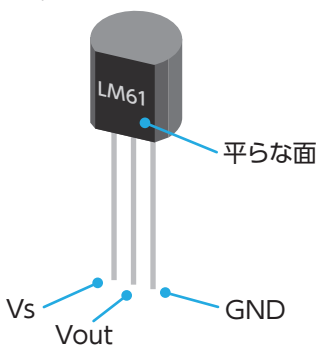


接続例





温度センサ



温度を計測するセンサです。
センサによって形状や出力形式は異なります。
今回使用する温度センサはLM61で、センサの出力はアナログ電圧です。

測定可能温度	−30 ～ +100℃ (LM61CIZ) −25 ～ +85℃ (LM61BIZ)
電源電圧	2.7 ～ 10V
出力電圧	10mV/℃
温度 0℃の時の出力電圧	600mV

②Studuino:bitとPCを接続します。

③温度センサのアナログ入力から取得したデジタル値を、1秒ごとにREPLに表示するプログラムを記します。
下記をテキストエディタに入力し、「実行」ボタンを押すと、REPLに0～1023の間の整数が表示されます。
手やヘアドライヤーなどを使って温度センサを温めると値が変化することを確認してください。値が変化しない場合はブレッドボードの接続が間違っていないか確認してください。
デジタル入出力・アナログ入力オブジェクトの機能の詳細は付録Cを参照してください。

```
from pystubit.board import pin2
import time
while True:
    val = pin2.read_analog()
    print(val)
    time.sleep(1)
```

ポートP2の値を読み取り、0（0V）から1023（3.3V）までの間の整数を返す。

```
6       time.sleep(1)
Artec Studuino:Bit MicroPython REPL
264
268
274
271
273
274
274
275
274
273
```

④アナログ入力から取得したデジタル値を温度に変換し、Studuino:bitのLEDディスプレイに表示するプログラムを記します。displayオブジェクトの機能詳細は付録Dを参照してください。

ondo1.py

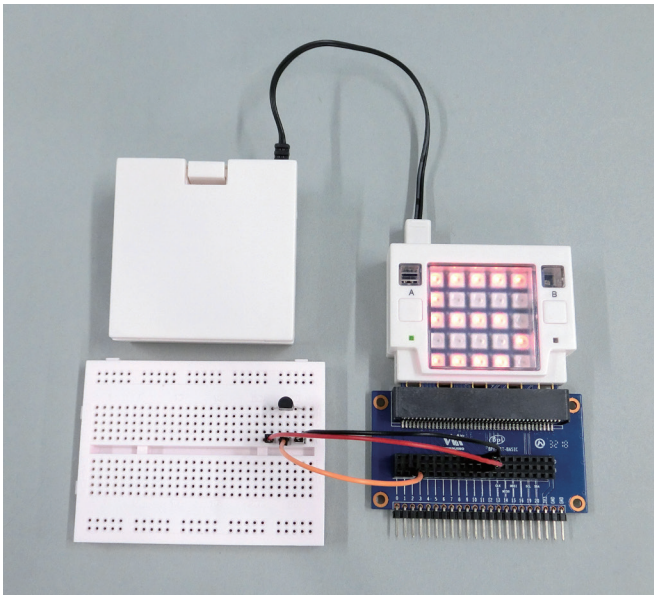
```
from pystubit.board import pin2, display
while True:
    val =int(330*pin2.read_analog()/1023 - 60)
    display.scroll(str(val)+'C')
```

センサ値を温度に変換（P.19の式③参照）

LEDディスプレイに表示

上記プログラムを実行してStuduino:bitのLEDディスプレイに温度（例:25C）がスクロール表示されることを確認してください。

Cは摂氏温度（℃）の代替文字です。
℃はLEDディスプレイに表示できないため、代替文字Cを使用しています。



⑤Muエディタの機能を使い、温度をグラフ表示するプログラムを記します。

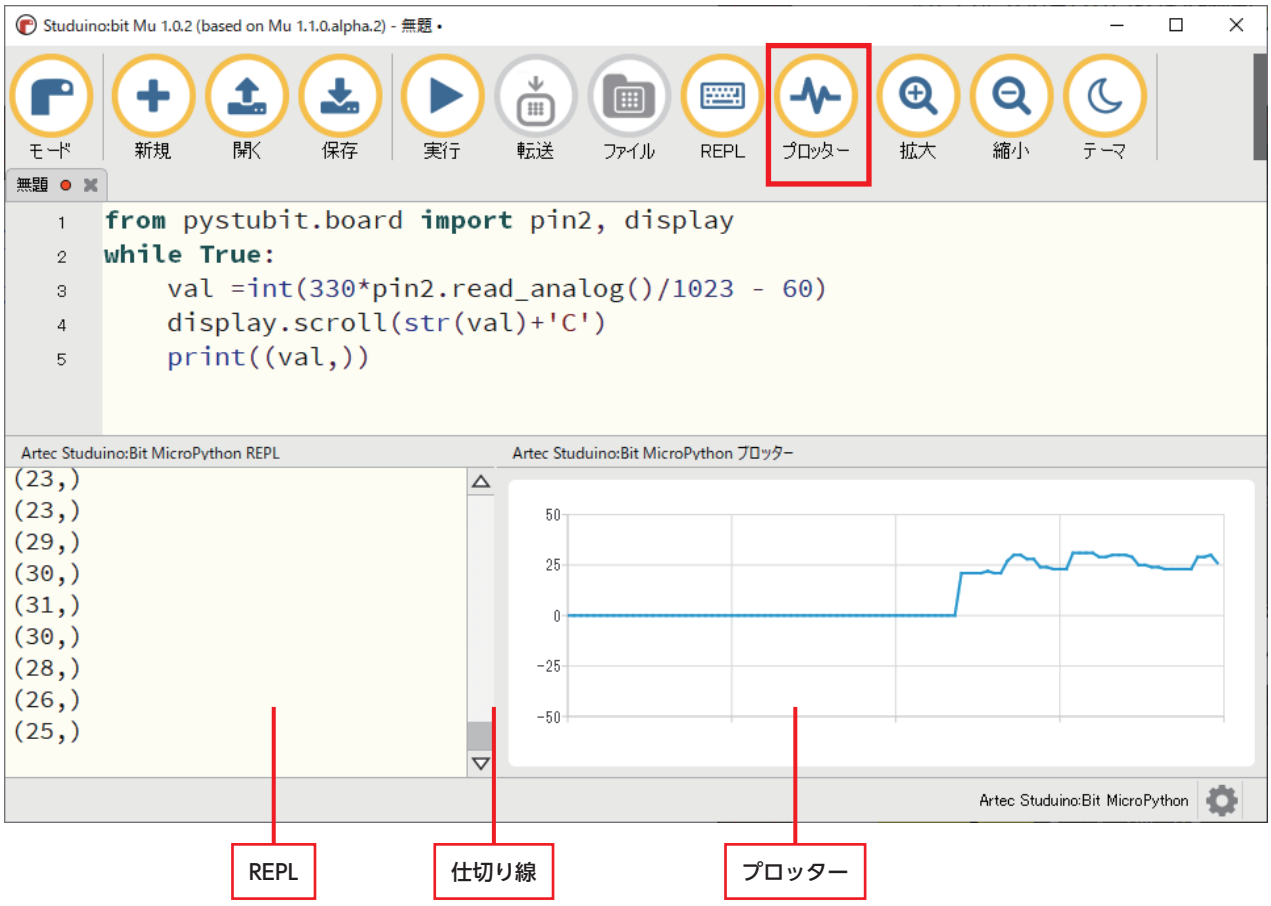
```
from pystubit.board import pin2, display
while True:
    val =int(330*pin2.read_analog()/1023 - 60)
    display.scroll(str(val)+'C')
    print((val,))
```

センサ値を温度に変換 (P.19の式③参照)

LEDディスプレイに表示

REPLとプロッターに表示

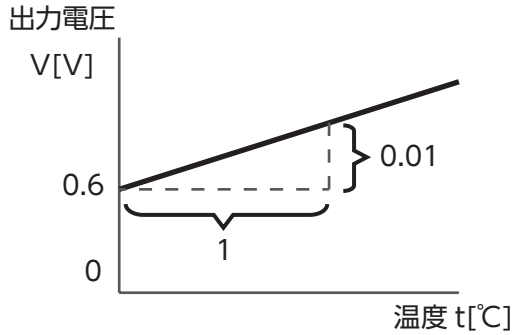
上記プログラムを実行すると、REPLに温度が表示されます。
メニューの「プロッター」ボタンをクリックすると、プロッターに温度がグラフ表示されます。
REPLとプロッターの間の仕切り線を左右にドラッグすると、それぞれの幅を変えることができます。
printの引数の「(val,)」は、タプル（一次元配列）で指定するため丸括弧()で囲み、要素が1個のタプルの末尾にはカンマ「,」が必要です。「print((val,))」を「print(val)」に変更すると、REPLには表示されますが、プロッターには表示されません。プロッターに表示するデータはタプルで指定する必要があります。
また、「print((val1, val2, val3))」のように指定すると、複数データのグラフを表示することもできます。



参考:温度とアナログ入力から取得したデジタル値の関係

温度センサの出力電圧Vと温度tの関係は式①のようになります。

$$V=0.01t+0.6$$ ……式①



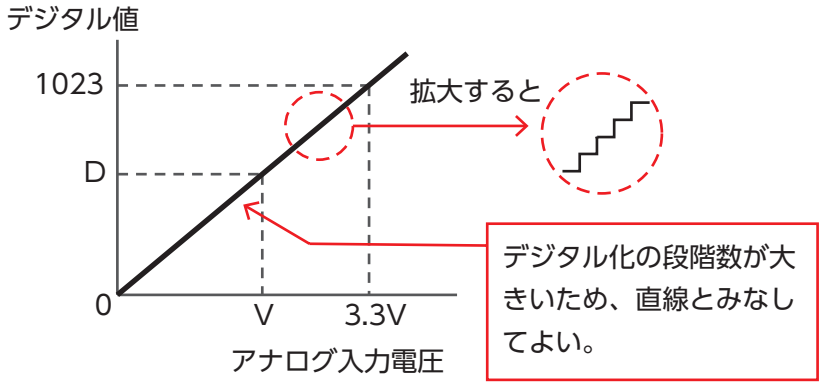
温度センサ LM61 の特性

山路を登りながら

山路を登りながら

ADコンバータのアナログ入力電圧Vとアナログ入力から取得したデジタル値Dの関係式は式②のようになります。

$$V=\frac{3.3D}{1023}$$ ……式②



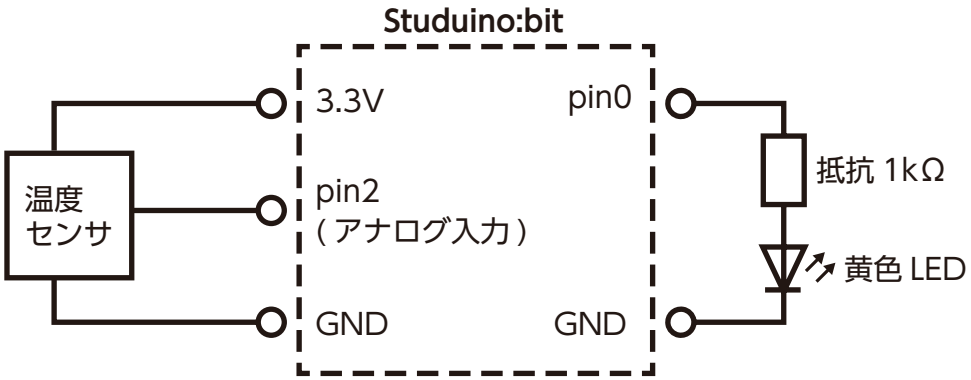
AD コンバータの特性

温度センサの出力電圧はADコンバータのアナログ入力電圧と等しいため、温度tとアナログ入力から取得したデジタル値Dの関係は式③と求まります。

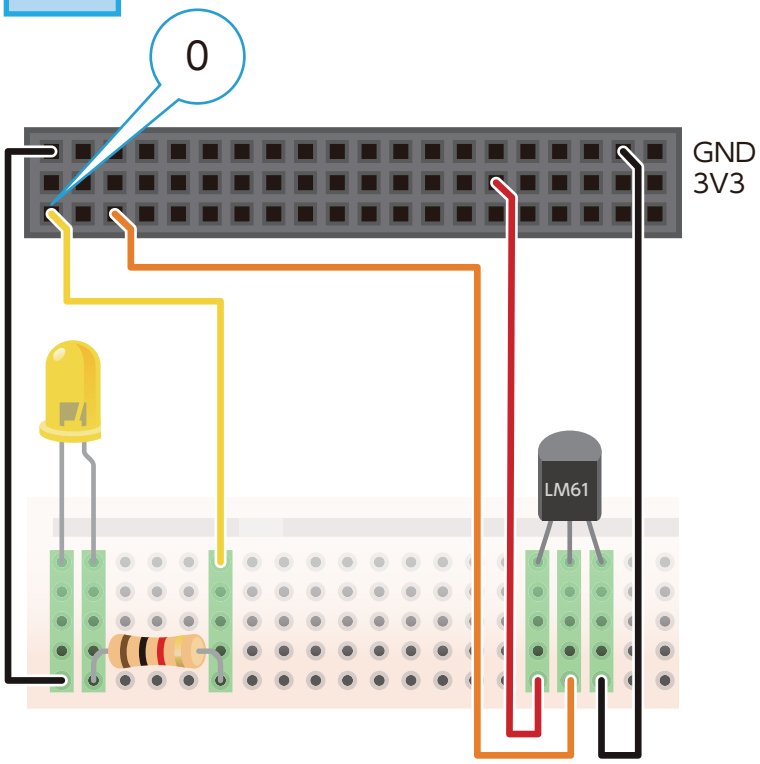
$$V=\frac{330D}{1023}-60$$ ……式③

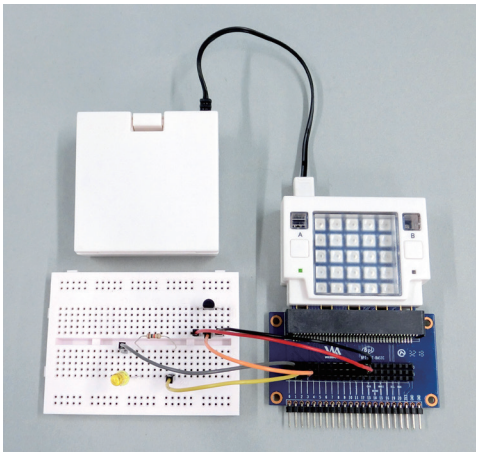
演習② 温度センサからのアナログ入力とLEDへのデジタル出力プログラム

計測した温度によってLEDの点灯と消灯を制御するプログラムを作成します。
①演習①のブレッドボードに黄色LEDと抵抗を追加します。

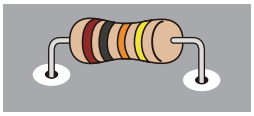


接続例



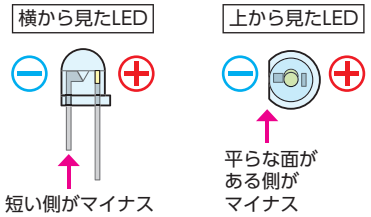


抵抗器



電流の流れを制限する部品です。抵抗の値は表面に記された4本の色帯で識別します。抵抗の値が大きくなるほど電流の流れる大きさは小さくなります。

LED



LED (Light Emitting Diode) はダイオードの一種です。電圧を与えることで発光する部品です。極性があるため、プラスとマイナスを間違えると発光しません。

②Studuino:bitとPCを接続します。
③黄色LEDを点灯するプログラムを記します。
　下記プログラムを転送して実行し、黄色LEDが点灯することを確認してください。
　LEDが点灯しない場合はブレッドボードの接続が間違っていないか確認してください。

```
from pystubit.board import pin0
pin0.write_digital(1)
```

ポート0に接続された黄色LEDを点灯する

④計測した温度によってLEDの点灯と消灯を制御するプログラムを記します。

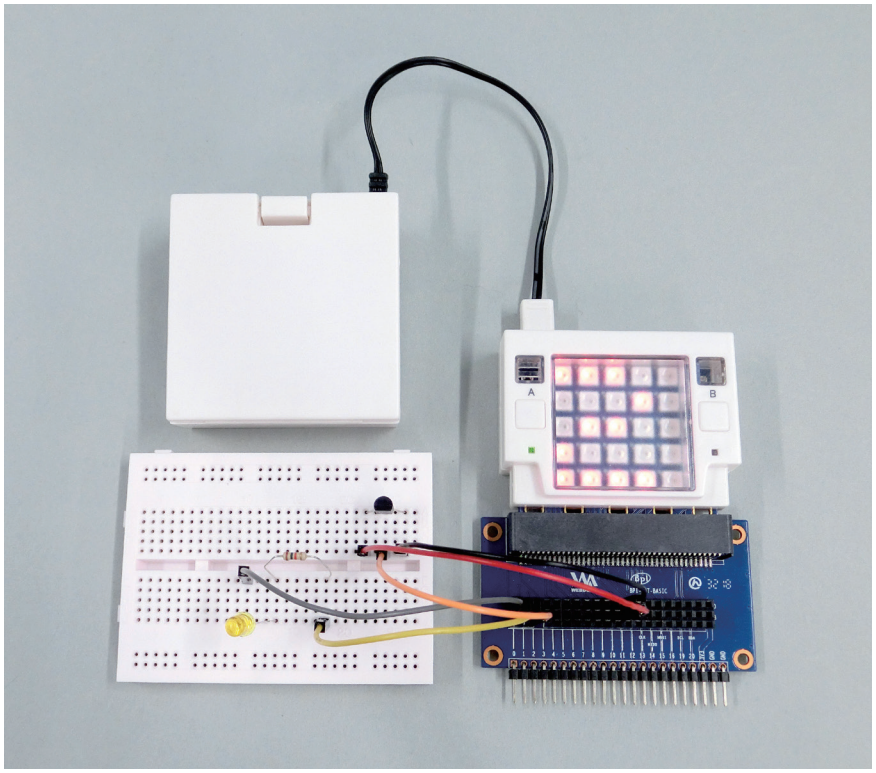
seigyo.py

```
from pystubit.board import pin0, pin2, display
while True:
    val =int(330*pin2.read_analog()/1023 - 60)
    display.scroll(str(val)+'C')
    if val >= 30:
        pin0.write_digital(1)
    else:
        pin0.write_digital(0)
```

30℃以上のときLEDを点灯する

30℃未満のときLEDを消灯する

上記プログラムを実行して、Studuino:bitのLEDディスプレイに温度がスクロール表示され、温度が30℃以上のときは黄色LEDが点灯、30℃未満のときは消灯することを確認してください。



演習③ 温度センサからのアナログ入力と2つのLEDへのデジタル出力プログラム

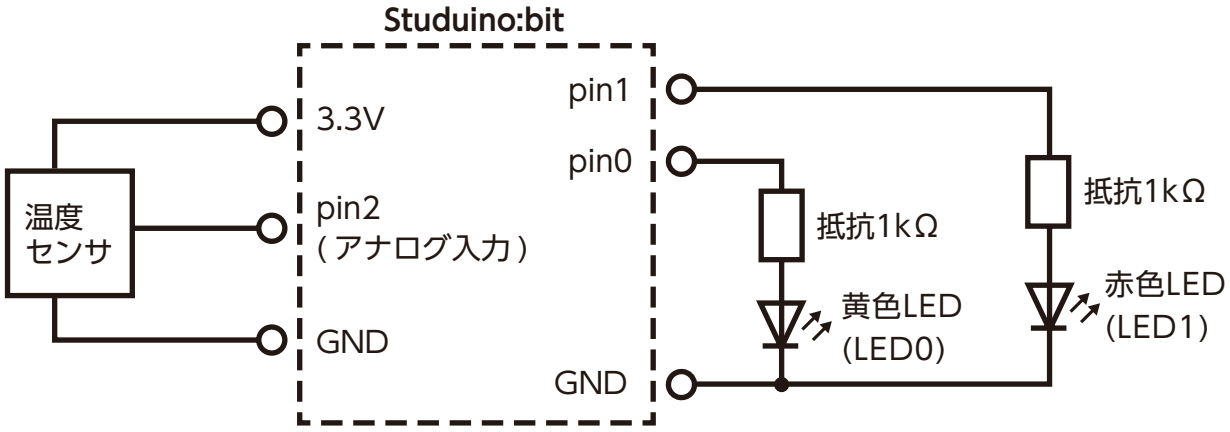
計測した温度によって2つのLEDの点灯と消灯を制御するプログラムを作成します。

条件	LED 0	LED 1
室温＜30℃	消灯	消灯
30℃≦室温＜35℃	点灯	消灯
室温≧35℃	消灯	点灯

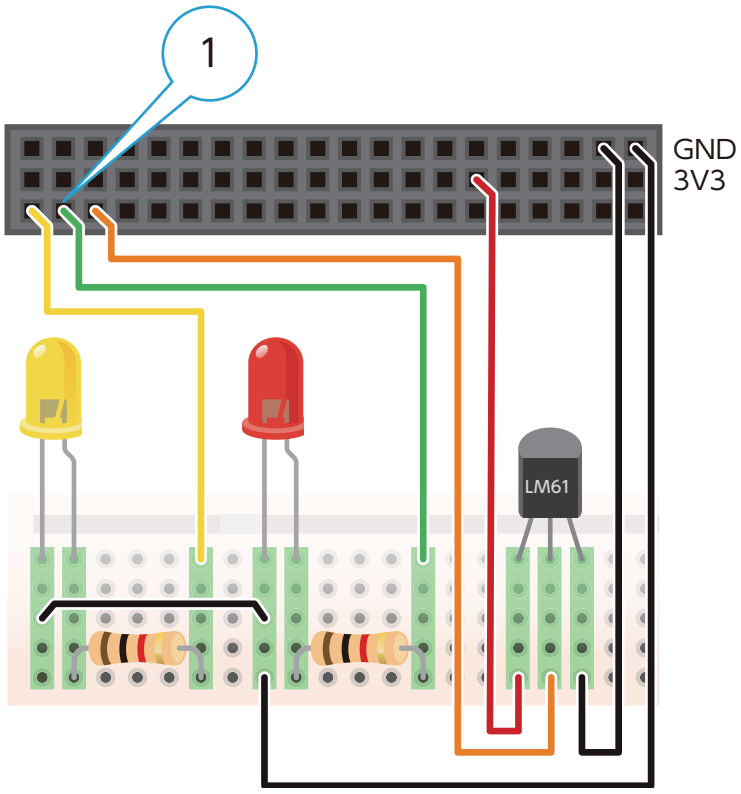
作例動画はコチラ

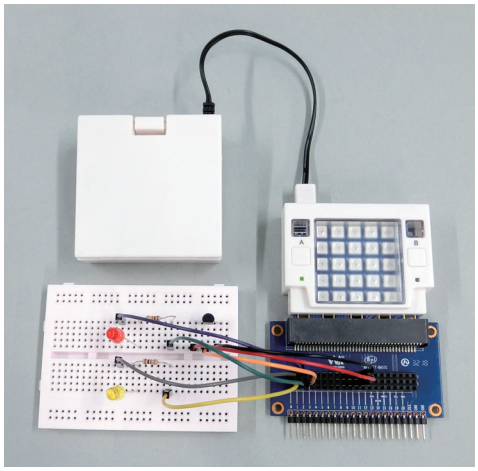
https://youtu.be/XBnDEUUzKbU

①演習②のブレッドボードに赤色LEDと抵抗を追加します。



接続例





②Studuino:bitとPCを接続します。
③赤色LEDを点灯するプログラムを記します。
　下記プログラムを転送して実行し、赤色LEDが点灯することを確認してください。
　LEDが点灯しない場合はブレッドボードの接続が間違っていないか確認してください。

```
from pystubit.board import pin1
pin1.write_digital(1)
```

ポート1に接続された赤色LEDを点灯する

④計測した温度によってLEDの点灯と消灯を制御するプログラムを記します。

seigyo1.py

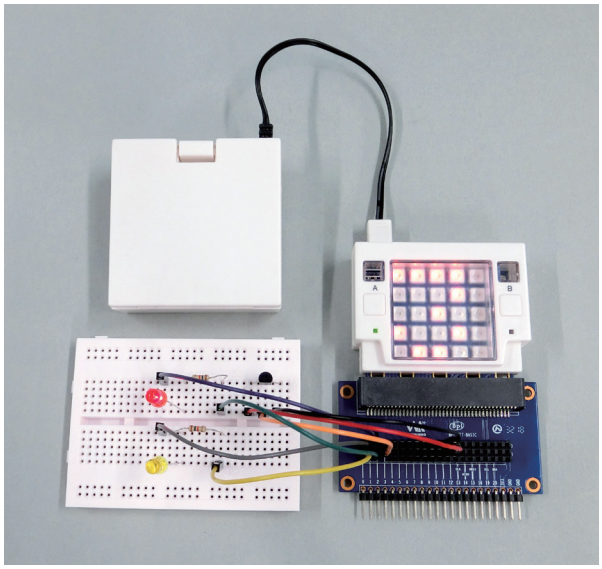
```
from pystubit.board import pin0, pin1, pin2, display
while True:
    val =int(330*pin2.read_analog()/1023 - 60)
    display.scroll(str(val)+'C')
    if val < 30:
        pin0.write_digital(0)
        pin1.write_digital(0)
    elif 30 <= val and val < 35:
        pin0.write_digital(1)
        pin1.write_digital(0)
    else:
        pin0.write_digital(0)
        pin1.write_digital(1)
```

30℃未満のとき
黄色LED、赤色LEDを消灯する

30℃以上35℃未満のとき
黄色LEDを点灯、赤色LEDを消灯する

35℃以上のとき
黄色LEDを消灯、赤色LEDを点灯する

上記プログラムを実行して、Studuino:bitのLEDディスプレイに温度がスクロール表示され、温度が30℃以上・35℃未満のときは黄色LEDが点灯し赤色LEDが消灯、35℃以上のときは黄色LEDが消灯し赤色LEDが点灯、30℃未満のときは2つのLEDが消灯することを確認してください。

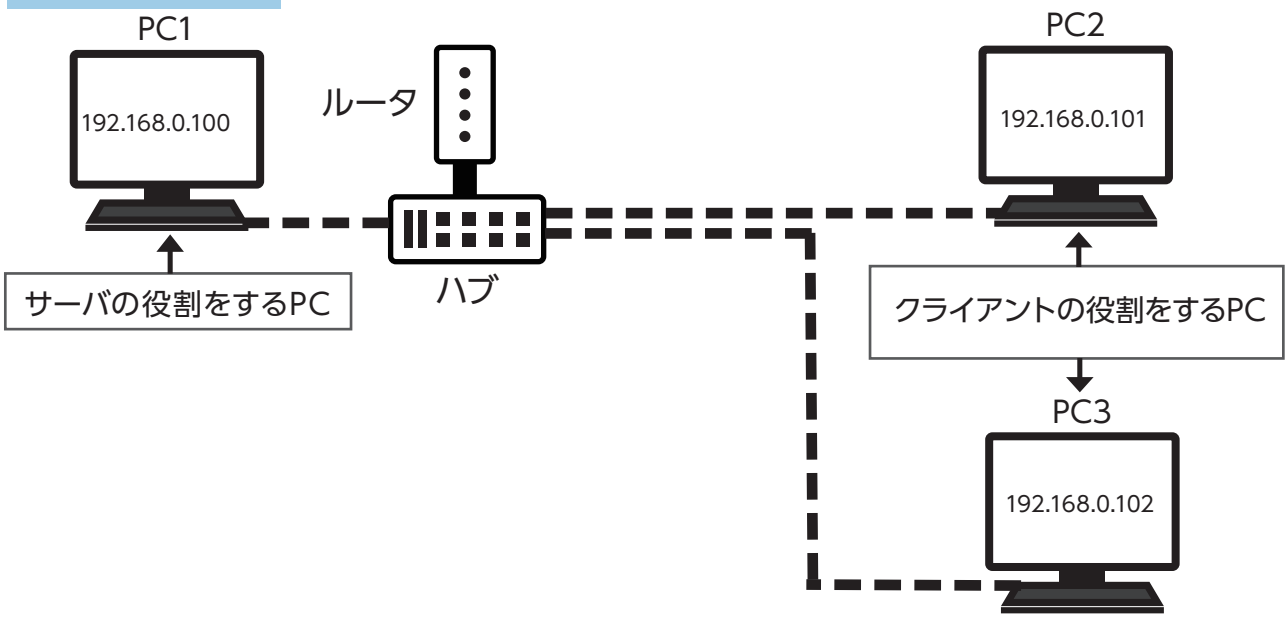


演習④ Webサーバの構築

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、コンピュータ教室など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

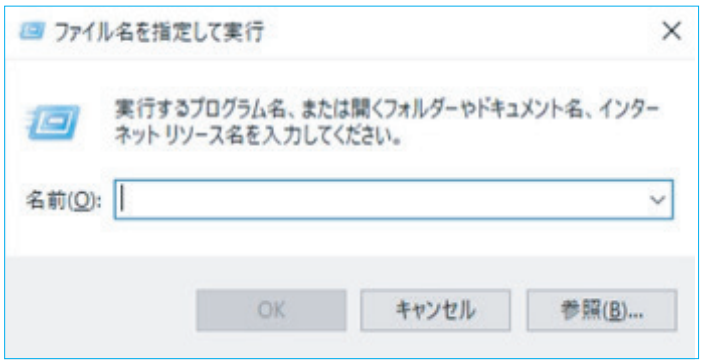


1.IP アドレスの確認

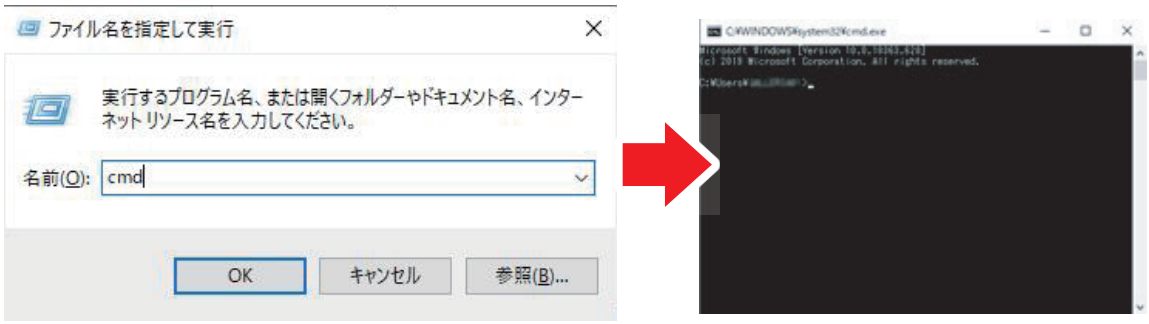
サーバの役割をする PC に割り当てられている IP アドレスを確認します。

Windows 環境

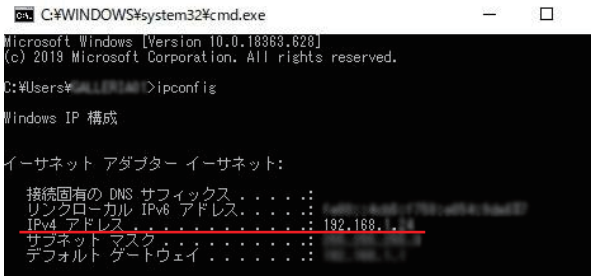
① 「Windows キー」+R キーを押してください。画面左下に下記のようなウインドウが表示されます。



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。

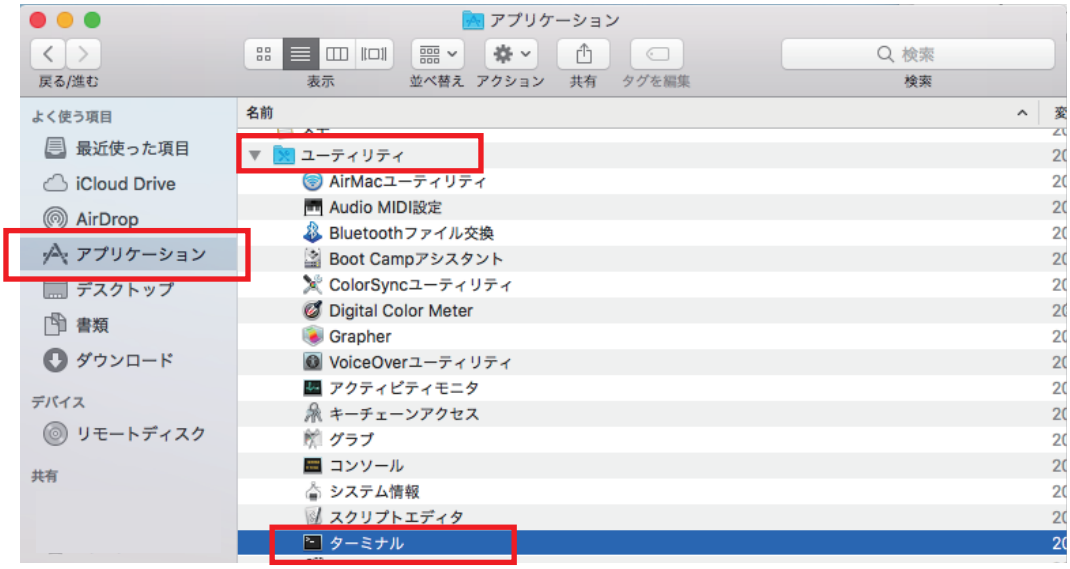


③ コマンドプロンプトに「ipconfig」と入力すると、IPアドレスを確認できます。



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。

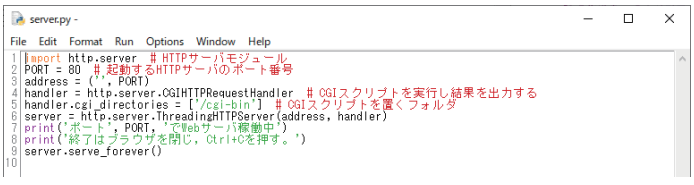
```
% ifconfig en0
en0: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MUI
        options=6463<RXCSUM,TXCSUM,TSO4,TSO6,CHANNEL_
        ether
        inet6 fe80::85d:c069:3b98:b7b0%en0 prefixlen
        inet 192.168.0.18 netmask 0xffffff00 broadcas
        nd6 options=201<PERFORMNUD,DAD>
        media: 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。

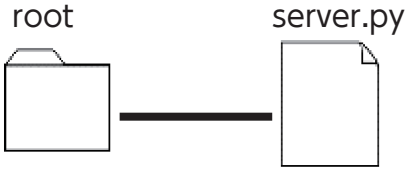
server.py

```
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）



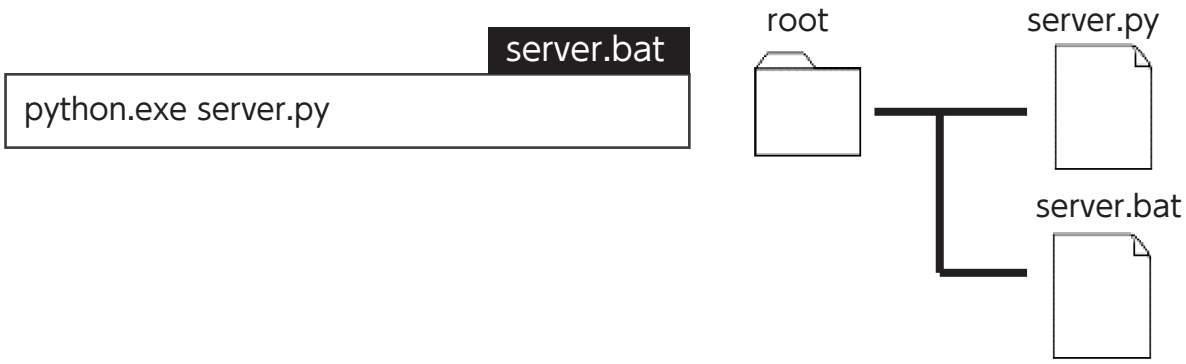
3. 自動実行ファイルの作成

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。

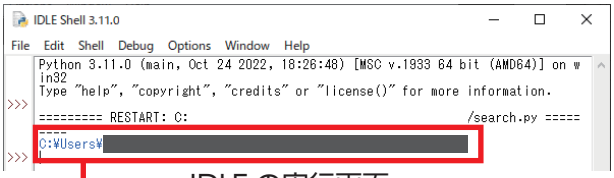
表示されたパス名(赤枠)を下記のserver.batのXXXXXXXに入力して、sever.batを作成します。

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。

search.py

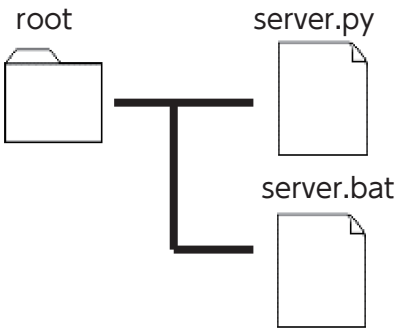
```
import sys
print(sys.exec_prefix)
```



IDLE の実行画面

server.bat

```
XXXXXXX¥python.exe server.py
```

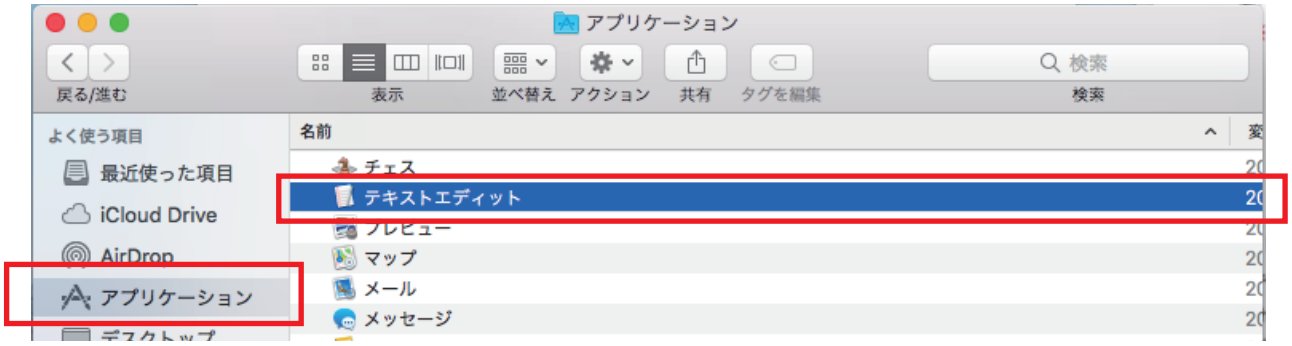


なお、P.3 でダウンロードしたプログラム内の server.bat はパス設定をした場合のファイルになっています。
パス設定をしなかった場合は server.bat を環境に合わせて修正してください。

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。

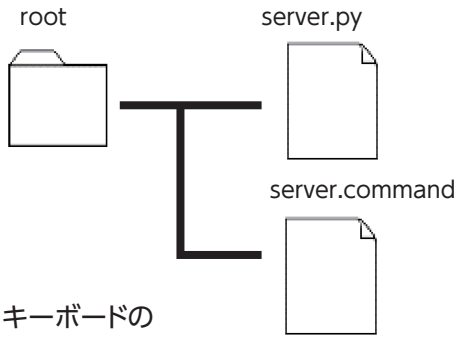


②テキストエディットを起動後、メニューの「フォーマット」をクリックして「標準テキストにする」を選択してください。

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

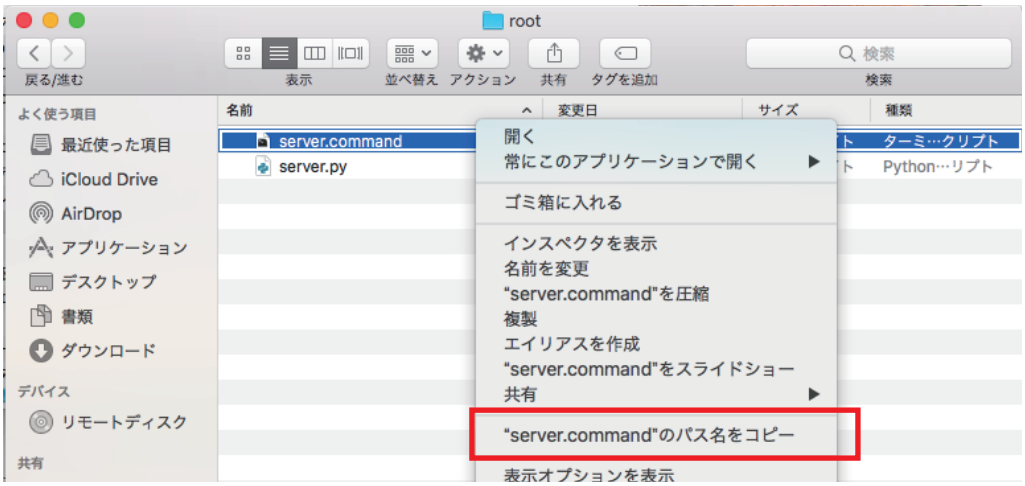
```
#!/bin/sh
cd `dirname $0`
python3 server.py
```



※「`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④「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でコピーして 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

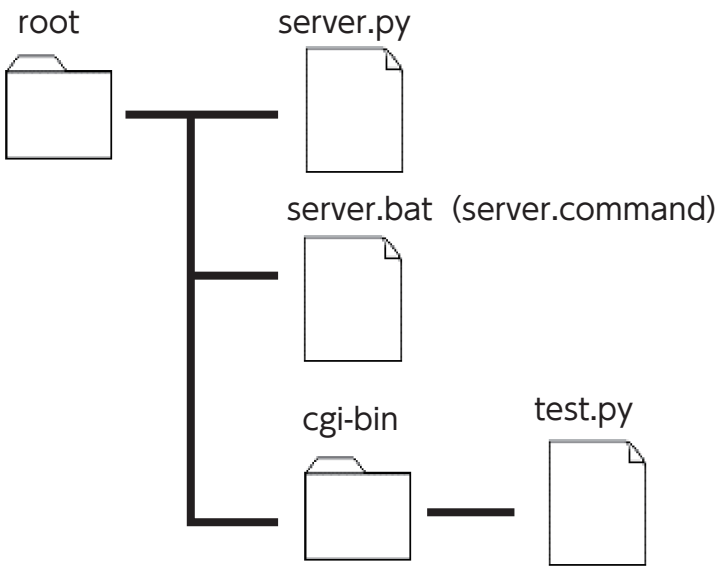
Windows 環境では基本的に Python の実行パスの記述は不要ですが、Mac 環境で実行する場合は、実行パスの記述が必要なため、py ファイルの先頭行に上記のコード「#!/usr/bin/env python3」を追加します。

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。

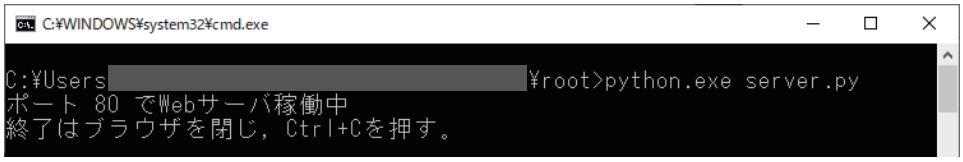
ただし、IDLE や VSCode 等の Python の実行環境で py ファイルを実行させる場合では、実行環境の中で処理系を指定しているため、シェバンの記述は不要です。

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）



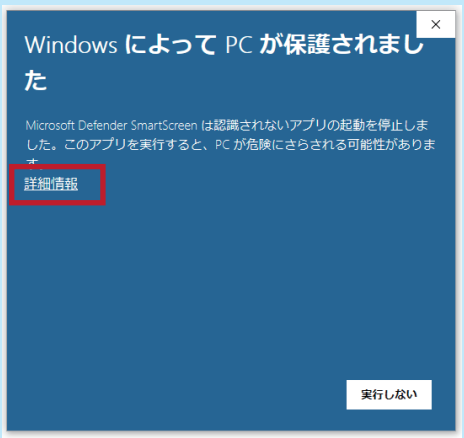
5. Web サーバの動作確認

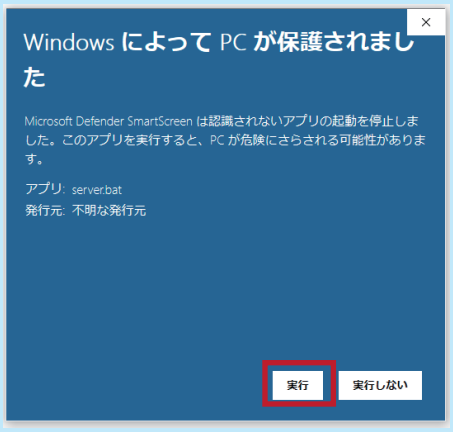
① sever.bat（Mac の場合は server.command）をダブルクリックすると、server.py が自動的に実行され、下記のような画面が表示されます。



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。

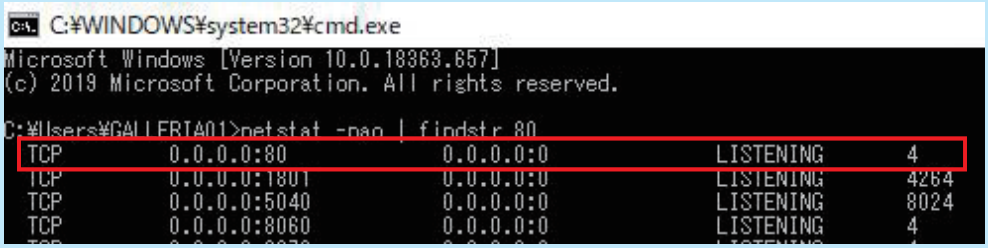
また、server.py（p.26）の2行目の「PORT=80」の部分を「PORT=8080」に変更する必要があります。

Windows 環境

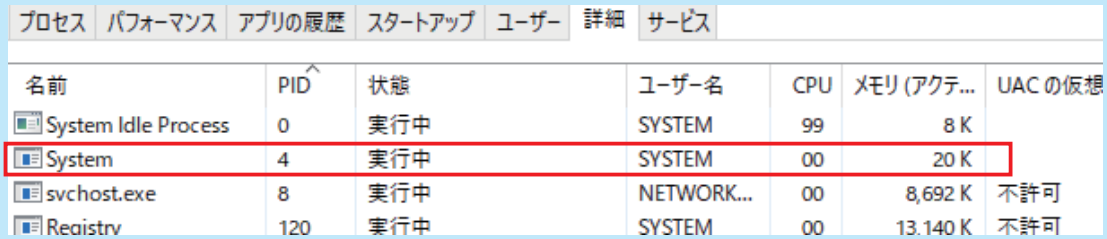
[ポートの使用状況の確認方法]

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



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。

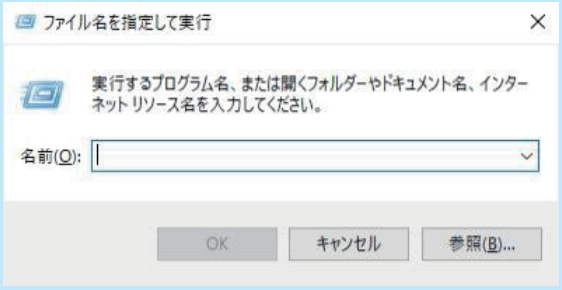


ポート番号80を使用しているサービスの一つとして、microsoftが提供しているIIS（Internet Information Services/インターネット インフォメーション サービス）というものがあります。前記のように「System」によってポート番号80を使用されている場合は、IISによるものの可能性があります。下記のようなアプリケーションソフトをインストールされているとIISが同時にインストールされ実行されている場合があります。

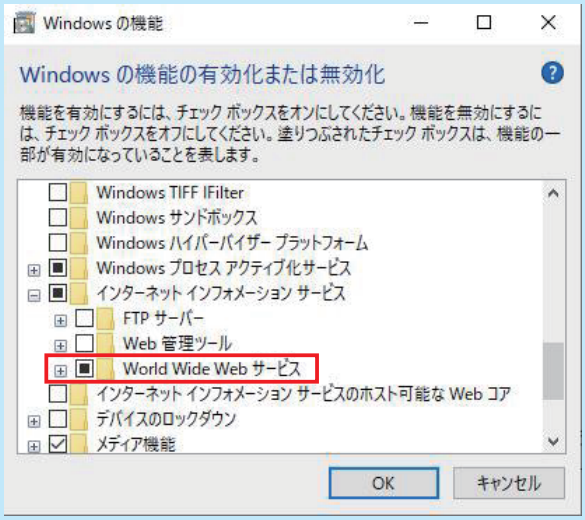
WebMatrix
Visual Studio
Windows Remote Management
World Wide Web Publishing Service

「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」ダイヤログを表示させ、「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＞

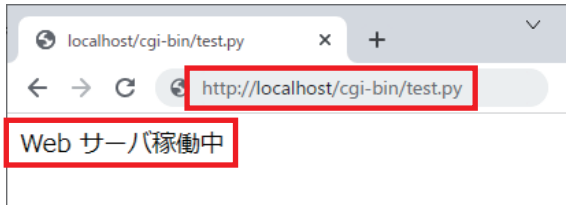
演習④1.IP アドレスの確認を参考にターミナルを起動して以下のように入力します。

lsof -P -i:80

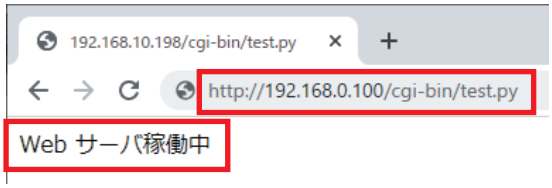
```
~ % lsof -P -i:80
COMMAND PID USER FD  TYPE    DEVICE SIZE/OFF NODE NAME
Python  370 user  4u IPv4 0xf3a63fd      0t0  TCP *:80 (LISTEN)
```

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。

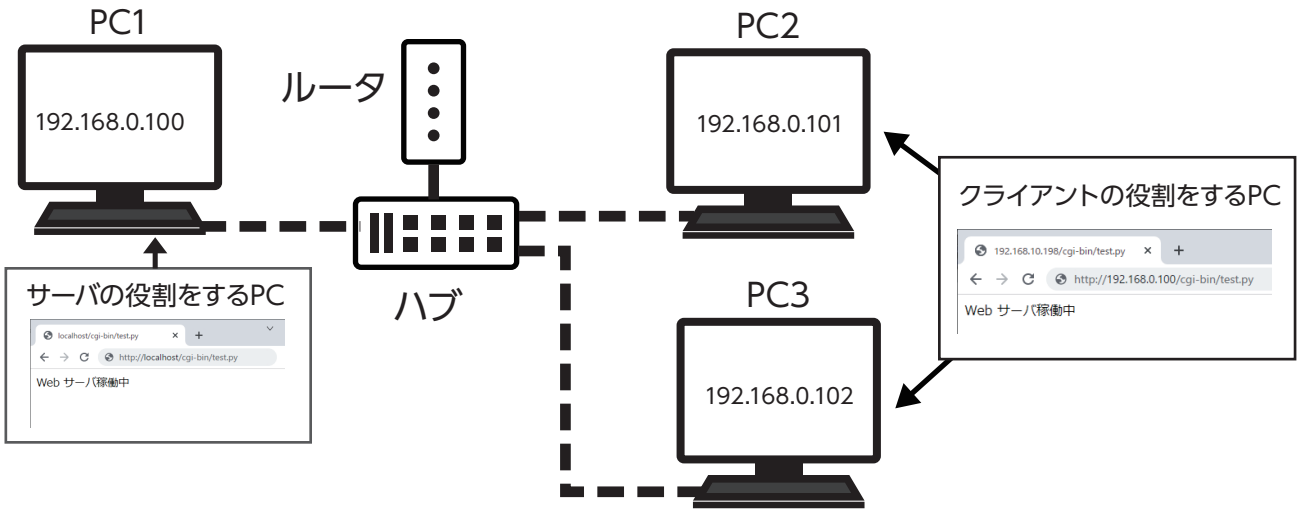


③LAN 接続された他の PC で Web ブラウザを開き、アドレス欄に「http://IP アドレス /cgi-bin/test.py」と入力し、同様に「Web サーバ稼働中」と表示されるか確認してください（※IP アドレス＝test.py が入っている Web サーバを起動した PC の IP アドレスを入力）。
80 以外のポート番号を指定した場合は「http://IP アドレス : ポート番号 /cgi-bin/test.py」と入力してください。



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。

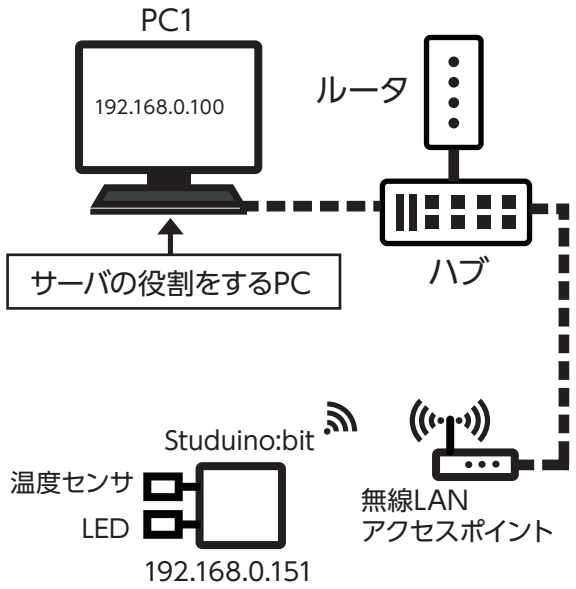


④Web サーバを終了する際は、①の画面で「Ctrl キー+C キー」を押します。
Windows 環境では続いて、「y」を入力して「Enter」を押してください。

演習⑤　Webサーバを用いた計測制御システムのモデルの構築

演習②で行った、室温が 30℃以上の場合に LED を点灯させ、30℃未満の場合に消灯させる条件分岐の分岐の部分の処理を Web サーバ側のプログラムで実行するシステムを作成します。
ここでは、Studuino:bit を Wi-Fi 経由でネットワーク接続し、Web サーバが制御します。

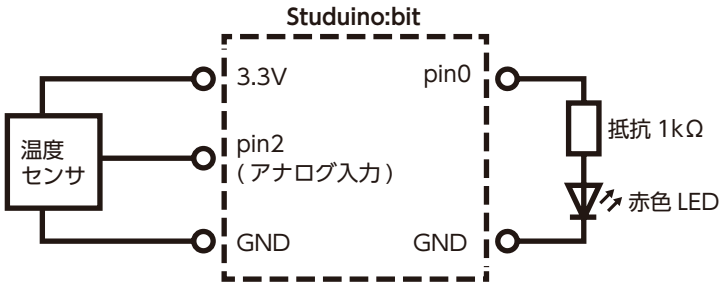
ネットワーク環境例　※IP アドレスの数値は一例



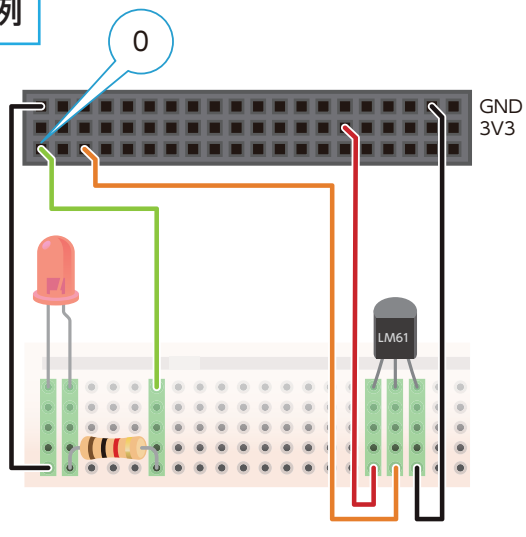
無線LAN アクセスポイントとしてWi-Fiルータを使用する場合は、Wi-Fiルータの取扱説明書を参照のうえ、ルータモード(RTモード)をOFFにして、アクセスポイントモード(APモード)に設定してください。
無線LANアクセスポイントとしてモバイルホットスポットを利用する場合は、付録Eを参照してください。
モバイルホットスポットとは、有線LANで接続されたWindows10やWindows11のコンピュータをWi-Fiルータ化する機能です。なお、図中のルータやハブ、PC1は、コンピュータ教室などに設置されている既存の装置を使用します。
ハブの空きポートに無線LANアクセスポイントをLANケー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

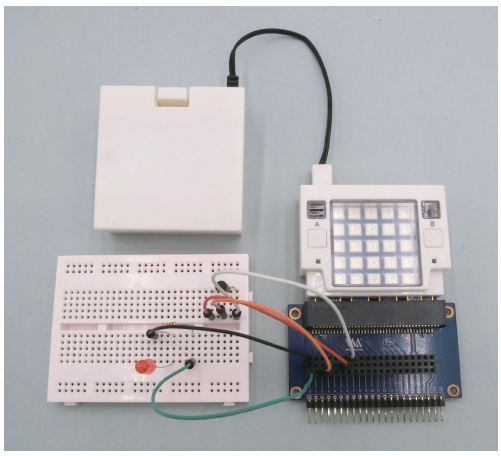
1.Studino:bit 側のプログラム作成

① Studuino:bitに赤色LED、抵抗、温度センサを接続します。



接続例





②Studuino:bitとPCを接続します。
③下記のsb.pyをMuエディタで作成し、Studuino:bitに転送します。
sb.pyはpin2から温度センサのデータを取得して、WebサーバにGETリクエストで送信します。Webサーバからはレスポンスとして文字列を受信し、ONである場合は赤色LEDが接続されているpin0に1を出力して点灯させ、OFFである場合は0を出力して消灯させます。
Wi-Fiオブジェクトの機能詳細は付録Fを参照してください。

sb.py

```
from pystubit.board import pin0, pin2
from pystubit_iot import wifi_config, wifi_connect, wifi_ifconfig, get_request
import time

# WiFi に接続する
wifi_config(ssid=' アクセスポイントの SSID', pwd=' パスワード ')
wifi_connect(trytime=5)
wifi_ifconfig(('Ⓐ', 'Ⓑ', 'Ⓒ', 'Ⓓ'))  # Studuino:bit に固定の IP アドレスを割り振る

# センサー値の送信先の指定
url = 'http://Web サーバの IP アドレス : ポート番号 /cgi-bin/device.py'

while True:
    time.sleep(1)
    val = int(330 * pin2.read_analog() / 1023 - 60)  # センサー値を取得

    # サーバにセンサー値を GET リクエストで送信
    # サーバからのデータ (ON/OFF) をレスポンスとして受信
    response = get_request(url, {'sens': str(val)})
    r = response.splitlines()

    # 受信したデータ情報から Studuino:bit の動きを設定する
    if r[0] == 'ON':
        pin0.write_digital(1)
    else:
        pin0.write_digital(0)
```

※ 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 ')
```

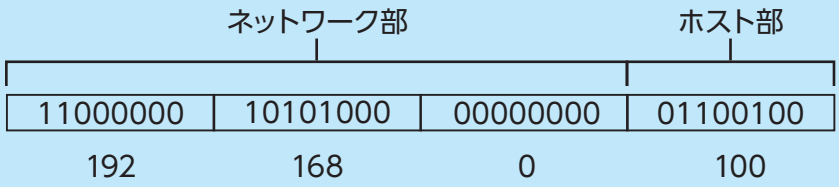
●wifi_ifconfig関数について

(1) IPアドレスのネットワーク部とホスト部

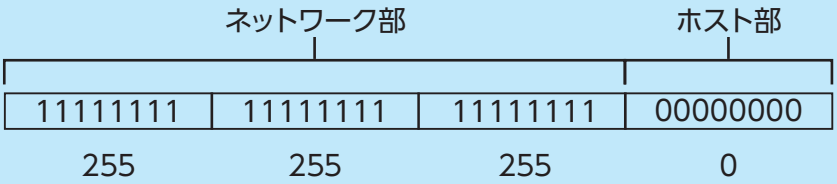
IPアドレスは、上位のネットワーク部と下位のホスト部から成り立っています。ネットワーク部は、そのIPアドレスが属しているセグメント(ネットワークの最小のまとまり)を識別するための部分で、ホスト部は、ネットワーク内のコンピュータを識別するための部分です。同一セグメント内ではネットワーク部はすべて等しくなります。例えば、256個のIPアドレスが利用できるクラスCでは、上位の24ビットがネットワークアドレス、下位の8ビットがホストアドレスになります。

<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(('Ⓐ','Ⓑ','Ⓒ','Ⓓ'))
```

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

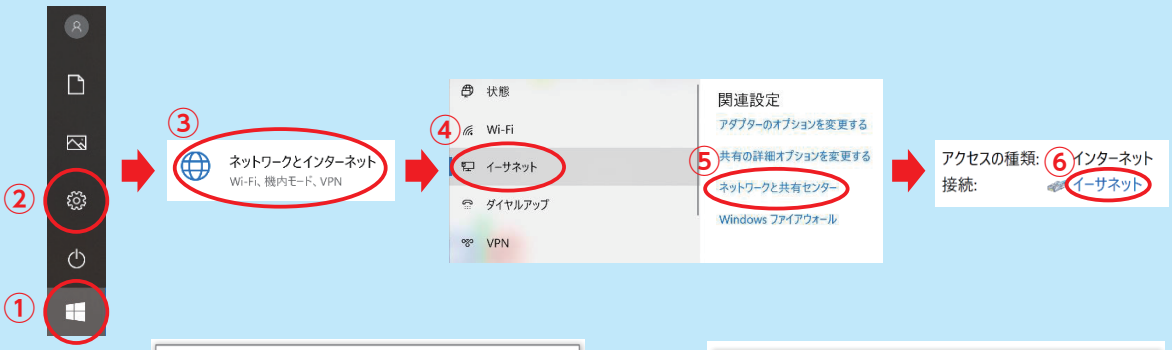
Ⓐ、Ⓑ、Ⓒ、Ⓓの各アドレスは以下のように設定します。

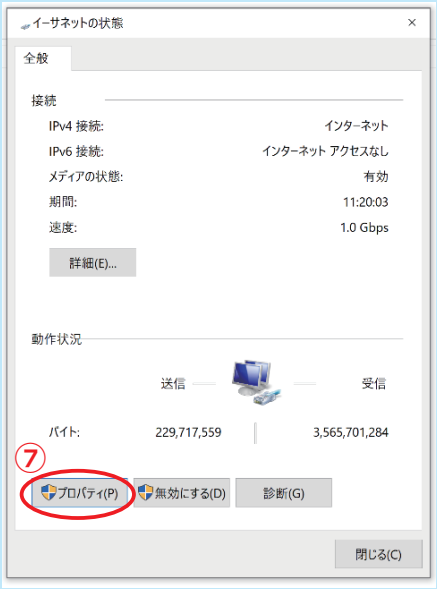
[コンピュータ教室の各アドレスの確認方法]

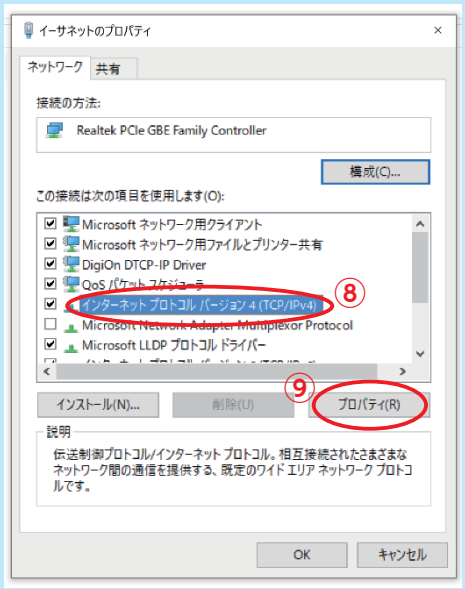
Windows 環境

コンピュータ教室のLAN内にある任意のPCを起動します。

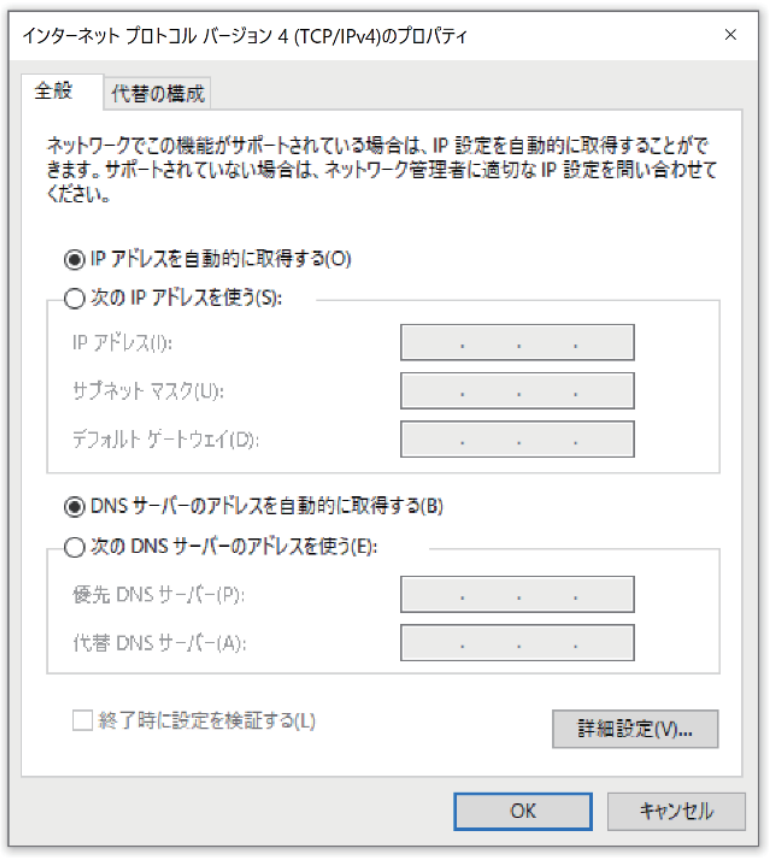
①スタートボタン → ②設定 → ③ネットワークとインターネット → ④イーサネット →⑤ネットワークと共有センター → ⑥イーサネット → ⑦プロパティ → ⑧インターネットプロトコルバージョン4(TCP/IPv4) → ⑨プロパティの順にクリックします。







※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。

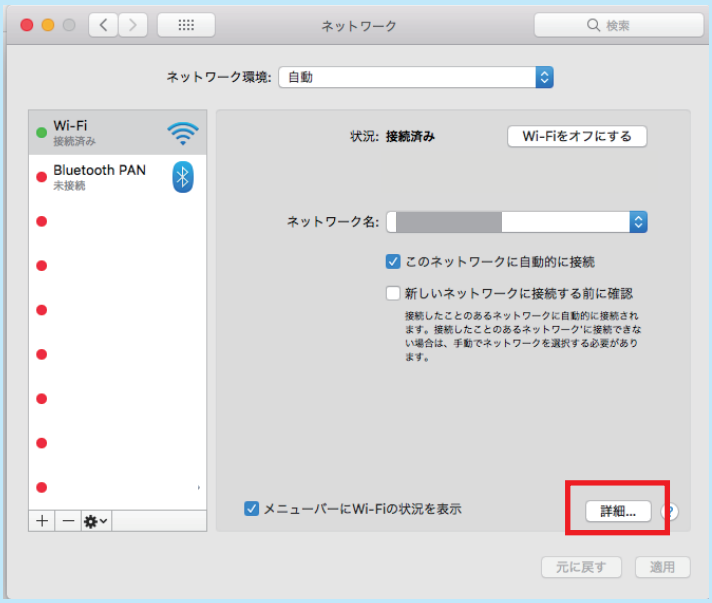


Mac 環境

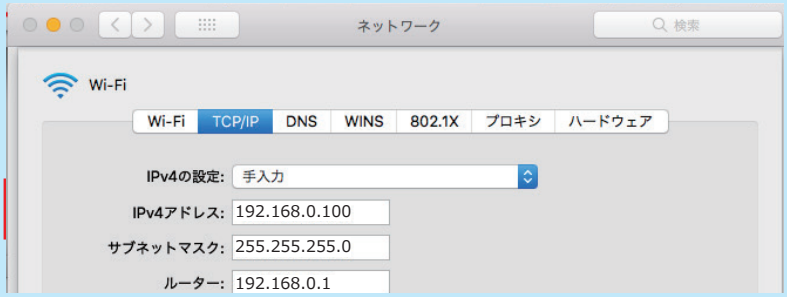
「Appleメニュー」（画面左上リンゴのマーク）-「システム環境設定」-「ネットワーク」をクリックします。



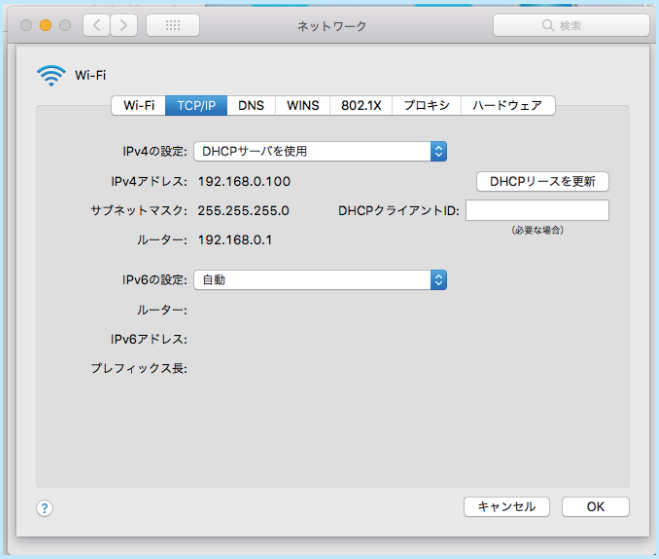
左側の項目から、コンピュータ教室の LAN に接続している「Wi-Fi」（無線 LAN）または、「Ethernet」(有線 LAN) を選択し、「詳細」ボタンをクリックします。



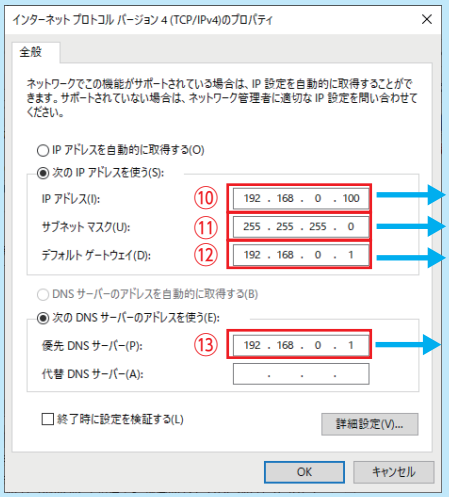
「TCP/IP」タブをクリックします。



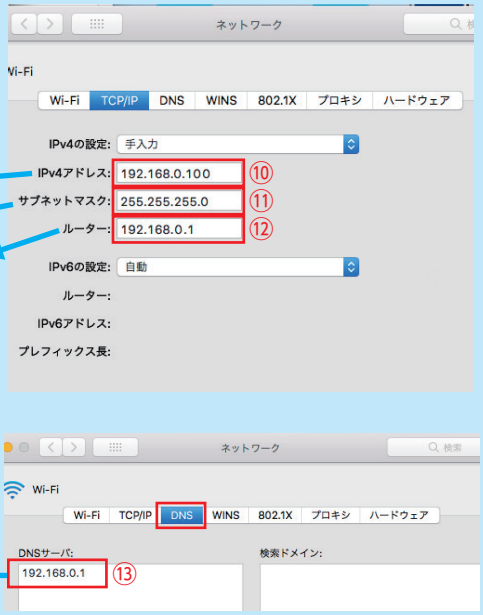
※以下のような画面が表示された場合は、LAN 内の装置には IP アドレスが自動で割り当てられています（自動で割り当てる仕組みを DHCP と呼びます）。このような場合は、プログラムの中の wifi_ifconfig 関数がある部分を削除してください。



Windows 環境



Mac 環境



Ⓐ: 下記参照
Ⓑ: サブネットマスク
Ⓒ: デフォルトゲートウェイ
Ⓓ: DNS サーバ

※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。

Ⓐ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.39の(2)のⒶ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。
Ⓐ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。
Ⓐ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)。

2. サーバ側のプログラム作成

①下記のdevice.pyをIDLEで作成します。

device.pyはStuduino:bitからの温度センサのデータを受信して、Studuino:bitへLEDのON/OFFの情報を送信するプログラムです。

※「¥」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。

※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Gを参照してください。

Windows 環境

device.py

```
import cgi
print('Content-Type: text/plain¥n')
param = cgi.FieldStorage()

val = param.getvalue('sens')
if int(val) >= 30:
    cont = 'ON'
else:
    cont = 'OFF'
print(cont)
```

Studuino:bit から送られたパラメータを取得

Studuino:bit の key(sens) から value(val) を得る

温度センサの値 (val) が 30℃以上の場合は ON、30℃未満の場合は OFF を cont に代入する

cont をレスポンスとして Studuino:bit に送信する

Mac 環境

device.py

```
#!/usr/bin/env python3
import cgi
print('Content-Type: text/plain¥n', end='¥r¥n')
param = cgi.FieldStorage()

val = param.getvalue('sens')
if int(val) >= 30:
    cont = 'ON'
else:
    cont = 'OFF'
print(cont, end='¥r¥n')
```

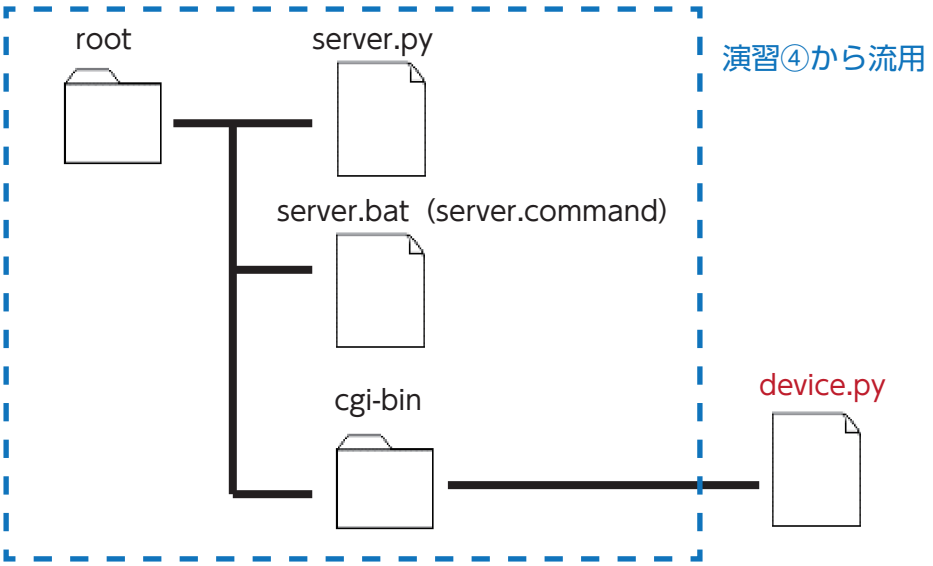
Studuino:bit から送られたパラメータを取得

Studuino:bit の key(sens) から value(val) を得る

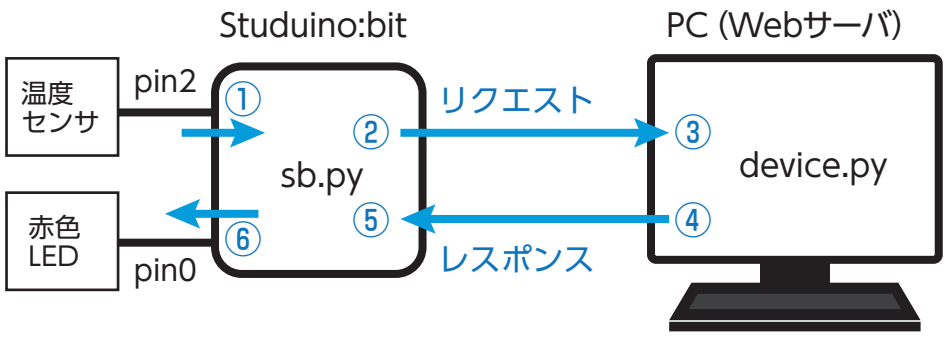
温度センサの値 (val) が 30℃以上の場合は ON、30℃未満の場合は OFF を cont に代入する

cont をレスポンスとして Studuino:bit に送信する

②演習④のtest.pyを削除し、cgi-binフォルダ下に①で作成したdevice.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sb.pyを実行すれば、sb.pyからdevice.pyが呼ばれ、sb.py⇔device.py間でデータの送受信が行われます。



全体の構成



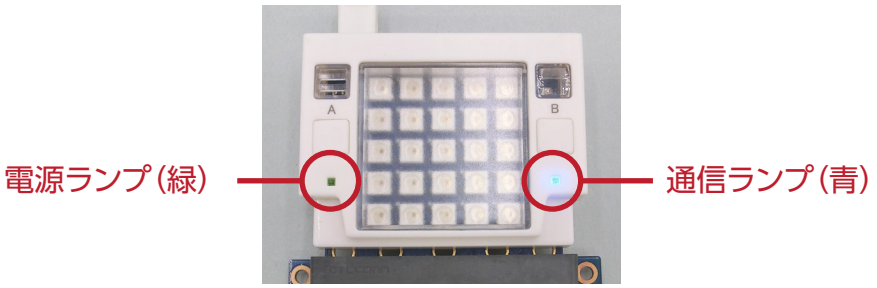
①	val = int(330 * pin2.read_analog() / 1023 - 60) 入力ポート pin2（温度センサ）から入力したデータを温度データに変換して val に代入する
②	response = get_request(url, {'sens': str(val)}) Web サーバ内の device.py に温度データ val を送信する
③	val = param.getvalue('sens') Studuino:bit から送信された温度データを取得し、val に代入する
④	print(cont) 温度データ val の値に応じた LED データ cont を Studuino:bit に送信する
⑤	response = get_request(url, {'sens': str(val)}) Web サーバから④で送信された LED データを②の戻り値として取得する
⑥	pin0.write_digital(1) / pin0.write_digital(0) LED データを出力ポート pin0（赤色 LED）に出力する

3. 動作確認

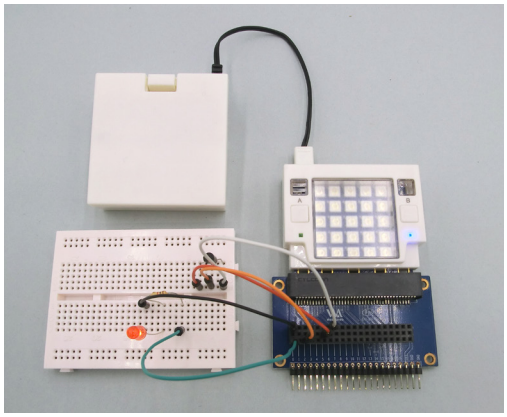
①sever.bat をダブルクリックして、Web サーバを起動します。



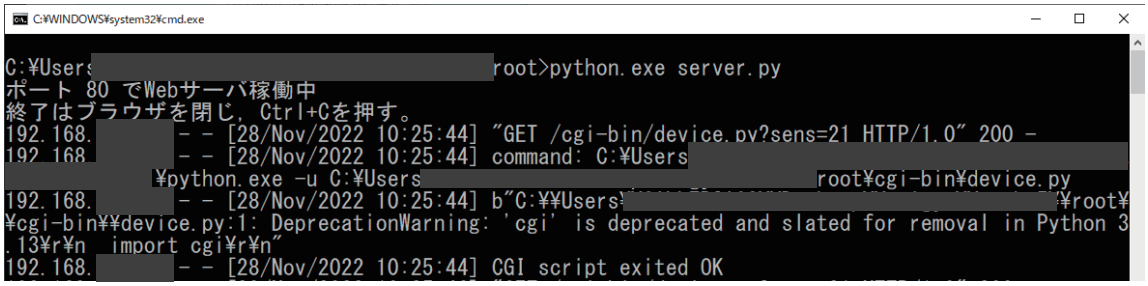
②Studuino:bitをPCから取り外し、電池ボックスまたはUSBケーブルで給電します。Studuino:bitのリセットボタンを押してsb.pyを実行します。電源ランプ（緑）と通信ランプ（青）が点灯していることを確認してください。
※以降の手順において、動作しない場合はStuduino:bitのリセットボタンを押して再起動を行ってください。



③温度が 30℃以上の時は赤色 LED が点灯、30℃未満の時は消灯することを確認してください。



Web サーバが Studuino:bit から温度データを取得し、LED データを送信すると、コマンドプロンプト (Mac 環境の場合はターミナル) が更新されます。



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」を押してください。

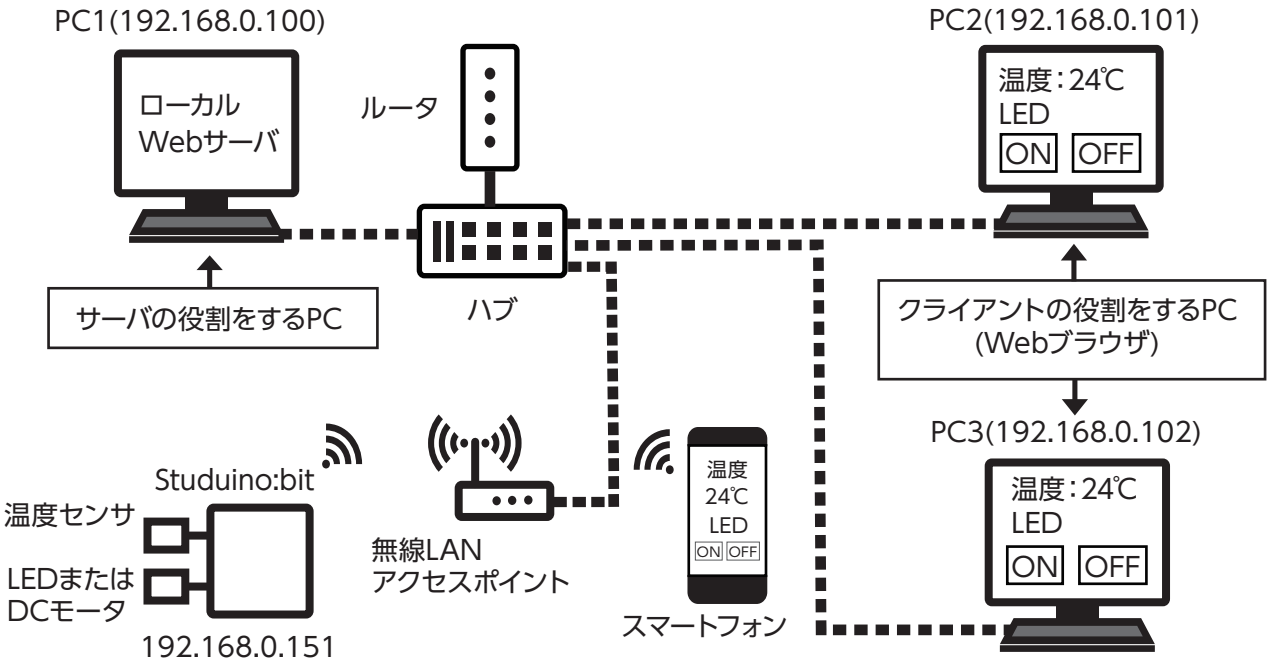
演習⑥　遠隔計測制御システムの構築

Web ブラウザ上からネットワーク経由で遠隔地の温度を計測したり、LED の ON/OFF を制御できるシステムを作成します。

作例動画はコチラ

https://youtu.be/tUOkzAvWS2Y

ネットワーク環境例　※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、PC2、PC3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。

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.Studino:bit 側のプログラム作成

①演習⑤のStuduino:bit、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sb1.pyをMuエディタで作成し、Studuino:bitに転送します。
演習⑤のsb.pyと内容は同じですが、データの送受信を行うサーバ側のプログラムのファイル名がdevice1.pyになるため、赤線下線のURLのみ変更しています。

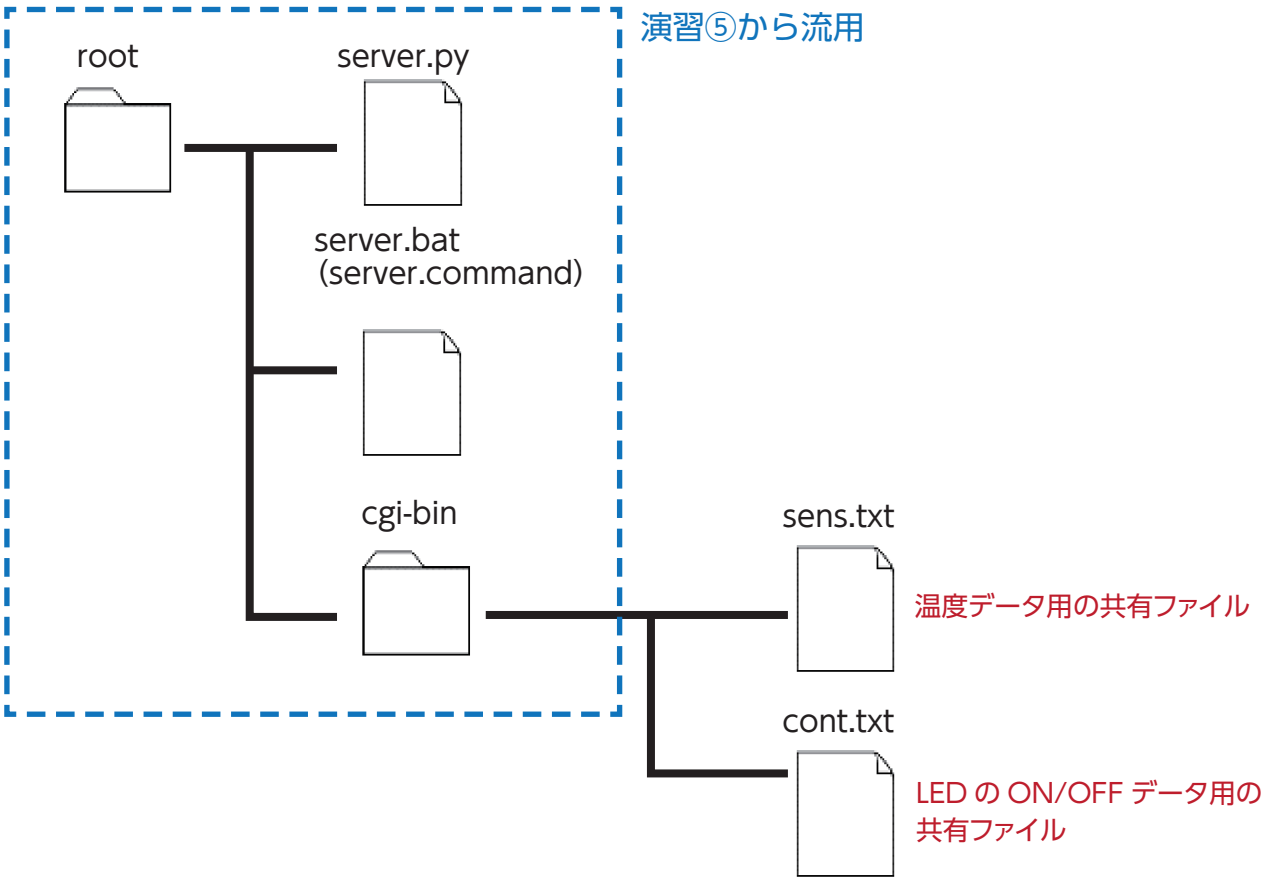
sb1.py

```
from pystubit.board import pin0, pin2
from pystubit_iot import wifi_config, wifi_connect, wifi_ifconfig, get_request
import time
wifi_config(ssid=' アクセスポイントの SSID', pwd=' パスワード ')
wifi_connect(trytime=5)
wifi_ifconfig(('Ⓐ', 'Ⓑ', 'Ⓒ', 'Ⓓ'))
url = 'http://Web サーバの IP アドレス : ポート番号 /cgi-bin/device1.py'
while True:
    time.sleep(1)
    val = int(330 * pin2.read_analog() / 1023 - 60)
    response = get_request(url, {'sens': str(val)})
    r = response.splitlines()
    if r[0] == 'ON':
        pin0.write_digital(1)
    else:
        pin0.write_digital(0)
```

2. サーバ側のプログラム作成

①演習⑤のdevice.pyファイルを削除します。

②Windows環境は「メモ帳」、Mac環境は「テキストエディット」などのテキストエディタを使って、「cgi-bin」フォルダ下に空の「sens.txt」ファイルと「cont.txt」ファイルを作成します。



③下記のindex.pyをIDLEで作成し、「cgi-bin」フォルダ下に格納します。
index.pyはsens.txtに保存されたデータをWebブラウザに送信し、Webブラウザから受信したデータをcont.txtに保存します。
Mac環境のみ先頭行に「#!/usr/bin/env python3」を追加してください。
※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Gを参照してください。

index.py

```
import cgi, sys, io

# ファイルにデータを書き込む関数
def fwrite(data, fn):
    file = open(fn, mode='w', encoding='utf-8')
    file.write(data)
    file.close()
# ファイルからデータを読み込む関数
def fread(fn):
    file = open(fn, mode='r', encoding='utf-8')
    text = file.readline()
    file.close()
    return text

# Web ブラウザに表示する HTML のコード
html = '''
<!DOCTYPE html>
<html lang="ja">
  <head>
    <meta charset="UTF-8">
    <title>IoT の実験 </title>
  </head>
  <body>
    <h2> 温度：{0}℃
      <input type="button" value=" 更新 "
       onclick="window.location.reload();">
      <form action="index.py" method="post">
        LED：
        <input type="submit" name="btn1" value="ON">
        <input type="submit" name="btn2" value="OFF">
      </form>
    </h2>
  </body>
</html>
'''
```

※次のページに続く

※前のページの続き

```
# utf-8 で標準出力を開き直すことで正しく utf-8 で出力されるようにする
sys.stdout = io.TextIOWrapper(sys.stdout.buffer, encoding='utf-8')
print('Content-Type: text/html; charset = utf-8¥n¥n')

# ファイル名
fsens = 'cgi-bin/sens.txt'  # Studuino:bit から受信したデータ
fcont = 'cgi-bin/cont.txt'  # ブラウザのボタンで設定した ON/OFF 情報
form = cgi.FieldStorage()   # ブラウザのボタンの状態を取得する
if form.getvalue('btn1') == 'ON':     # ON が押されている場合ファイルに ON を書き込む
    fwrite('ON', fcont)
elif form.getvalue('btn2') == 'OFF':  # OFF が押されている場合ファイルに OFF を書き込む
    fwrite('OFF', fcont)
val = fread(fsens)  # Studuino:bit から受信したデータを読み込む
print(html.format(val))   # ブラウザに読み込んだデータを反映した HTML コードを表示
```

④下記のdevice1.pyをIDLEで作成し、「cgi-bin」フォルダ下に格納します。device1.pyはcont.txtに保存されたデータをStuduino:bitに送信し、Studuino:bitから受信したデータをsens.txt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Gを参照してください。

Windows 環境

device1.py

```
import cgi

# ファイルにデータを書き込む関数
def fwrite(data, fn):
    file = open(fn, mode='w', encoding='utf-8')
    file.write(data)
    file.close()
# ファイルからデータを読み込む関数
def fread(fn):
    file = open(fn, mode='r', encoding='utf-8')
    text = file.readline()
    file.close()
    return text

print('Content-Type: text/plain¥n')

# ファイル名
fsens = 'cgi-bin/sens.txt'  # Studuino:bit から受信したデータ
fcont = 'cgi-bin/cont.txt'  # ブラウザのボタンで設定した ON/OFF 情報

param = cgi.FieldStorage()      # Studuino:bit からの送られたパラメータを取得
val = param.getvalue('sens')    # パラメータの sens に設定されている値を取得
if val != None:
    fwrite(val, fsens)      # sens の値をファイルに書き込む
cont = fread(fcont)     # ブラウザの ON/OFF 設定を読み込む
print(cont)     # ON/OFF 情報を Studuino:bit に送信
```

Mac 環境

device1.py

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

# ファイルにデータを書き込む関数
def fwrite(data, fn):
    file = open(fn, mode='w', encoding='utf-8')
    file.write(data)
    file.close()
# ファイルからデータを読み込む関数
def fread(fn):
    file = open(fn, mode='r', encoding='utf-8')
    text = file.readline()
    file.close()
    return text

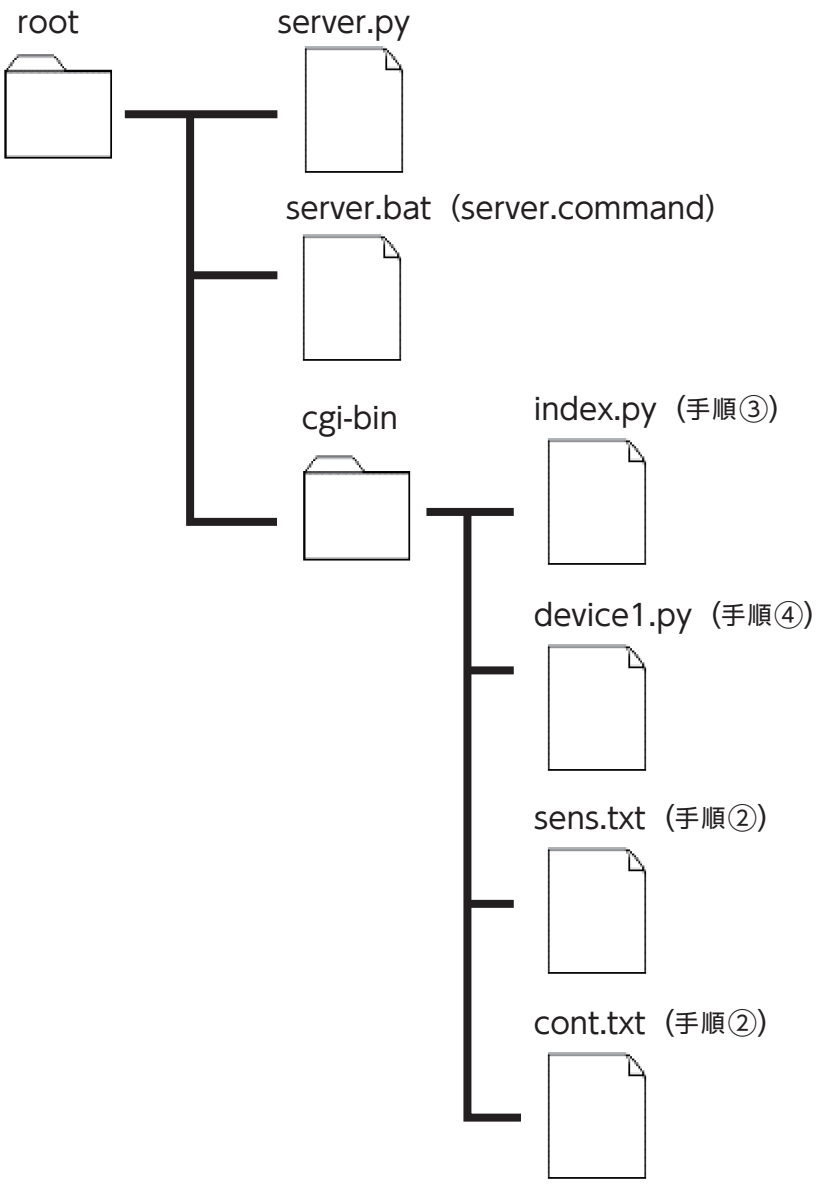
print('Content-Type: text/plain¥n', end='¥r¥n')

# ファイル名
fsens = 'cgi-bin/sens.txt'  # Studuino:bit から受信したデータ
fcont = 'cgi-bin/cont.txt'  # ブラウザのボタンで設定した ON/OFF 情報

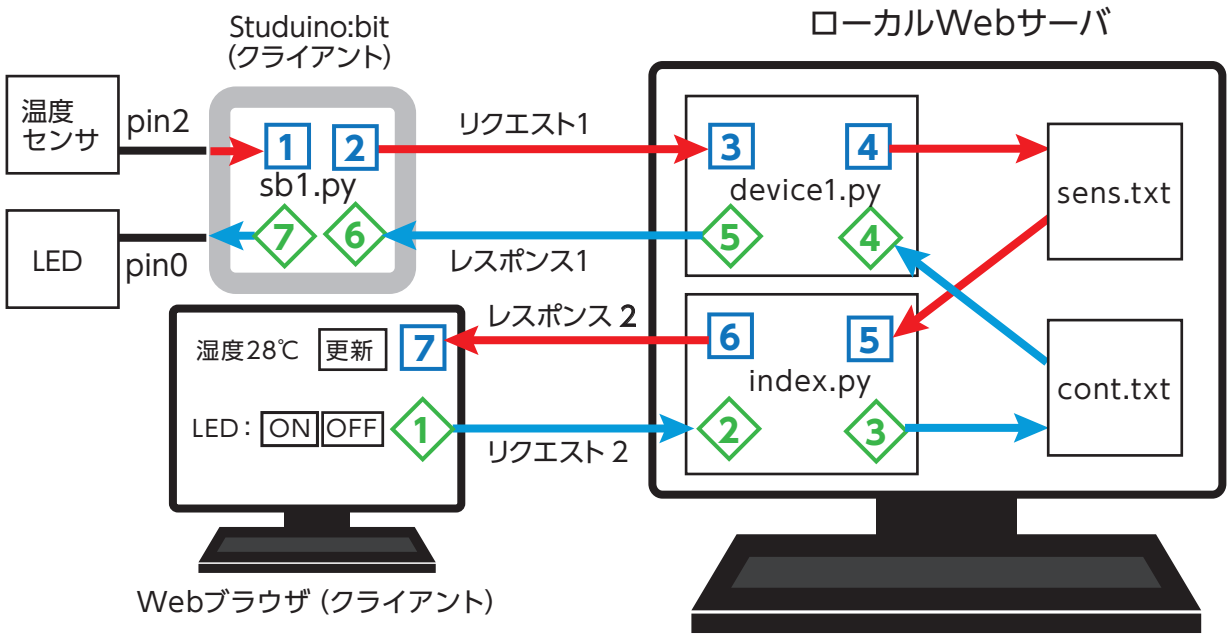
param = cgi.FieldStorage()      # Studuino:bit からの送られたパラメータを取得
val = param.getvalue('sens')    # パラメータの sens に設定されている値を取得
if val != None:
    fwrite(val, fsens)      # sens の値をファイルに書き込む
cont = fread(fcont)     # ブラウザの ON/OFF 設定を読み込む
print(cont, end='¥r¥n')   # ON/OFF 情報を Studuino:bit に送信
```

※「¥」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。

サーバ側のフォルダ構成は下記の通りです。



全体の構成



温度センサに関するメソッド

1	val = int(330 * pin2.read_analog() / 1023 - 60) 入力ポート pin2（温度センサ）から入力したデータを温度データに変換して val に代入する
2	response = get_request(url, {'sens': str(val)}) Web サーバ内の device1 .py に温度データ val を送信する
3	val = param.getvalue('sens') Studuino:bit から送信された温度データを取得し、val に代入する
4	fwrite(val , fsens) sens.txt に温度データ val を書き込む
5	val = fread(fsens) sens.txt から読み出した温度データを変数 val に代入する
6	print(html.format(val)) レスポンスとしてクライアントの Web ブラウザに val を送信する
7	<h2> 温度：{0}℃ クライアントの Web ブラウザに温度データを表示する

LED に関するメソッド

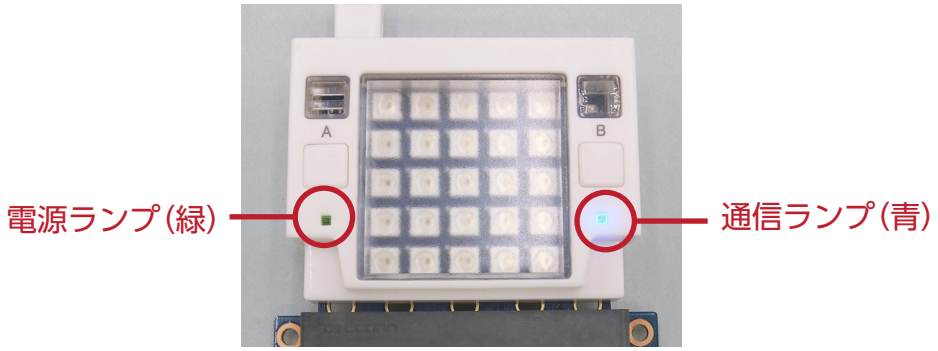
①	<form action="index.py" method="post"> Web サーバ内の index.py に LED データ（ON・OFF）をリクエストとして送信する
②	form.getvalue('btn1') / form.getvalue('btn2') Web ブラウザからの LED データを取得する
③	fwrite('ON',fcont) / fwrite('OFF',fcont) cont.txt に LED データを書き込む
④	cont=fread(fcont) cont.txt から LED データを読み出し、変数 cont に代入する
⑤	print(cont)　　【Mac 環境】print(cont, end='¥r¥n') LED データをレスポンスとして Studuino:bit に出力する
⑥	response = get_request(url, {'sens': str(val)}) Web サーバから ⑤ で送信された LED データを 2 の戻り値として取得する
⑦	pin0.write_digital(1) / pin0.write_digital(0) ポート pin0 に LED データを出力する

3. 動作確認

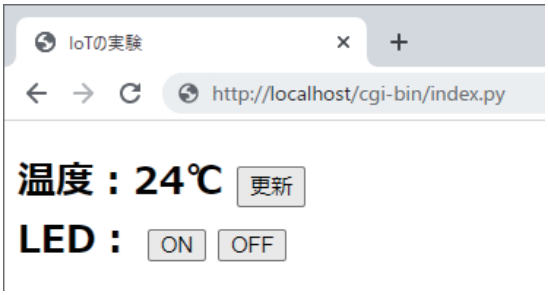
① sever.bat をダブルクリックして、Web サーバを起動します。



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sb1.pyを実行します。
電源ランプ（緑）と通信ランプ（青）が点灯していることを確認してください。
※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。



③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/index.py」と入力することで、下図のように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/index.py」
と入力してください。

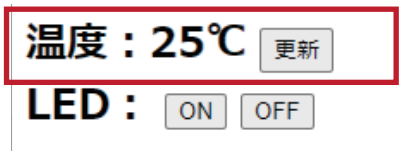


④LAN 接続された他の PC で Web ブラウザを開き、アドレス欄に
「http://IP アドレス /cgi-bin/index.py」
と入力することで、下図のように表示されることを確認してください。
（※IP アドレス＝test.py が入っている Web サーバを起動した PC の IP アドレスを入力）
80 以外のポート番号を指定した場合は
「http://IP アドレス : ポート番号 /cgi-bin/index.py」
と入力してください。

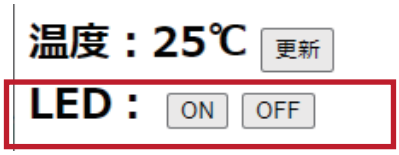


※IP アドレスの数値は一例

⑤更新ボタン押すと、現在の温度センサの値を取得できることを確認してください。



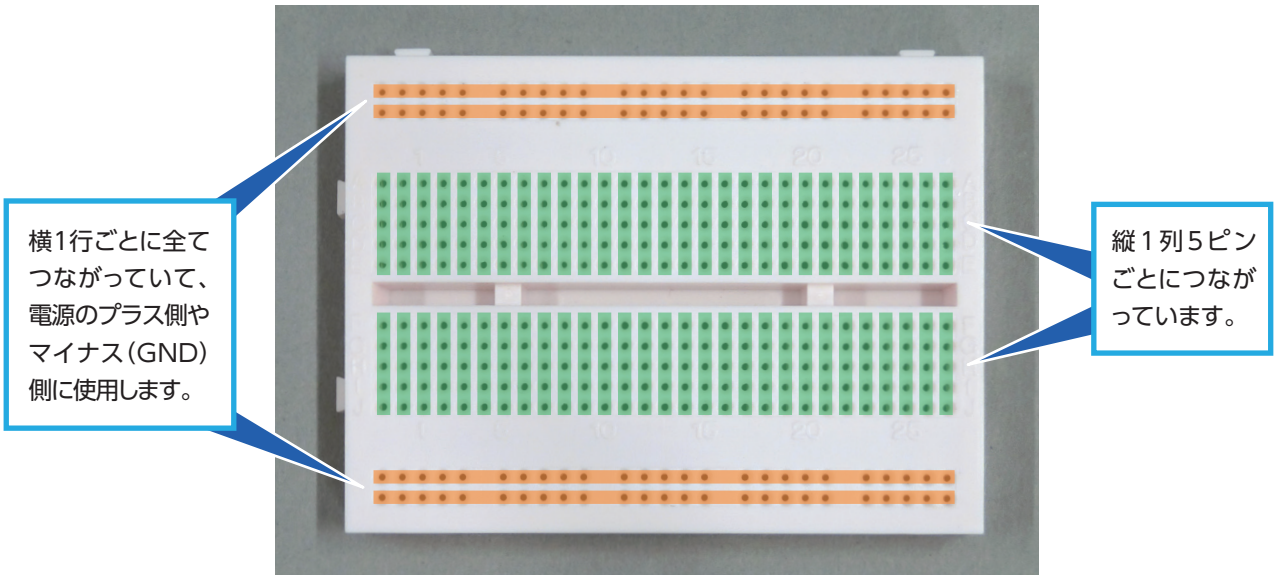
⑥ON ボタンを押すと、LED が点灯し、OFF ボタンをおすと LED が消灯することを確認してください。



⑦Web 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」を押してください。

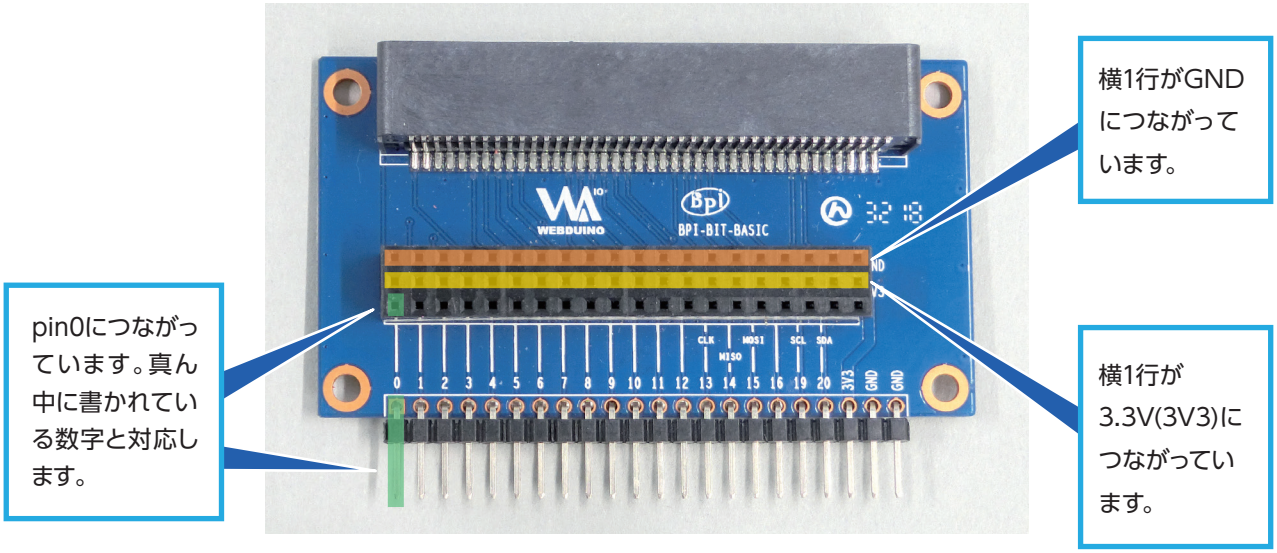
付録 A. ブレッドボードの使い方

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。



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。



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. デジタル入出力・アナログ入力オブジェクトの機能一覧

端子モジュールはpin0/pin1/pin2/pin3です。pin2/pin3はデジタル入出力に対応していないため、pin2.write_digital(value)やpin3.write_digital(value)を実行するとエラーになります。
デジタル入力用のread_digitalメソッドは使用できます。

メソッド	説明
write_digital(value)	value引数が1の場合はHighに、0の場合はLowにデジタル信号を設定します。
read_digital()	デジタル信号を取得します。High の場合は 1 を、Low の場合は 0 を返します。
write_analog(value)	PWM信号を端子に出力します、value引数は0(0%)～1023(100%)です。
status()	PWM の使用状況を表示します。
read_analog()	端子の電圧を読み取り、0(0V)から1023(3.3V)までの間の整数値を返します。

付録D. displayオブジェクトの機能一覧

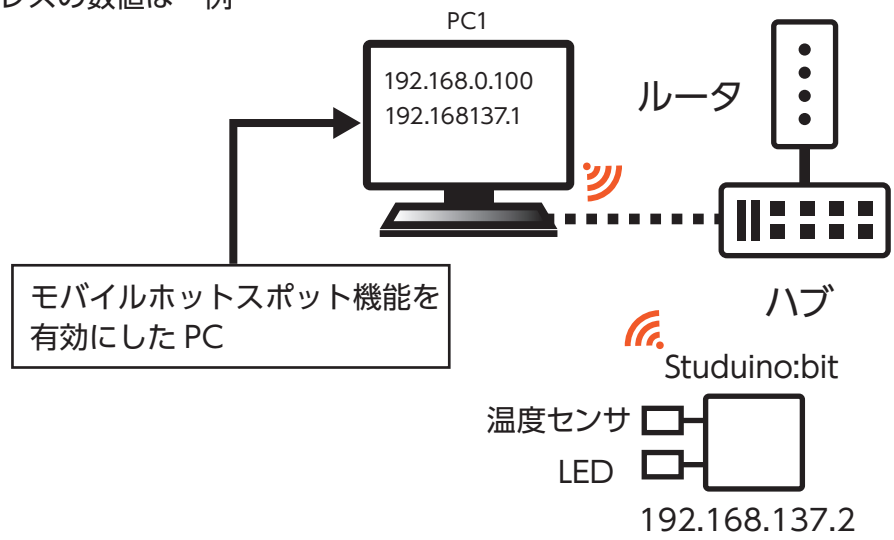
メソッド	説明
get_pixel(x, y)	x 列 y 行の LED の色を返します。返り値は (R,G,B) です。
set_pixel(x, y, color)	x 列 y 行の LED の色を color で設定します。color は、(R,G,B), [R,G,B], #RGB で指定します。
clear()	すべての LED の明るさを 0 (オフ) に設定します。
show(iterable, delay=400, *, wait=True, loop=False, clear=False, color=None)	iterable(イメージ / 文字列 / 数字) を順番に表示します。
scroll(string, delay=150, *, wait=True, loop=False, color=None)	string(文字 / 数字) を水平方向にスクロールさせます。
on()	ディスプレイの電源を ON にします。
off()	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)。
is_on()	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 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。



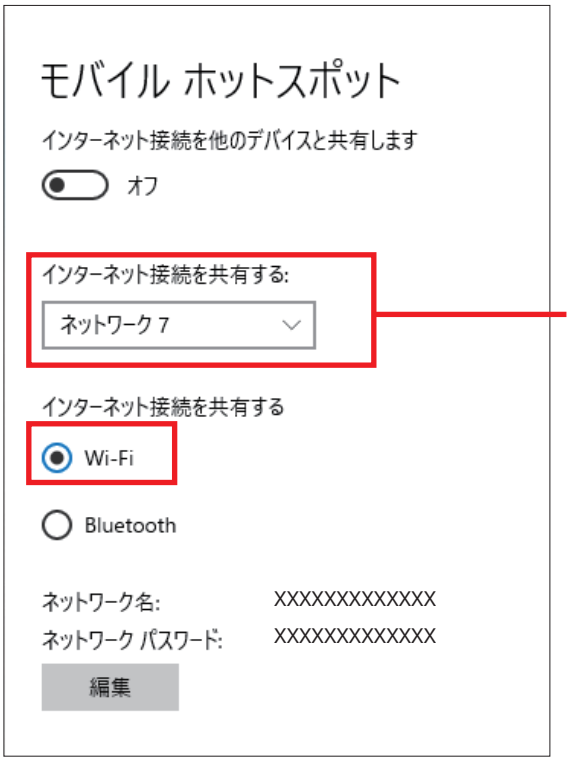
②「ネットワークとインターネット」をクリックします。



③「モバイルホットスポット」をクリックします。

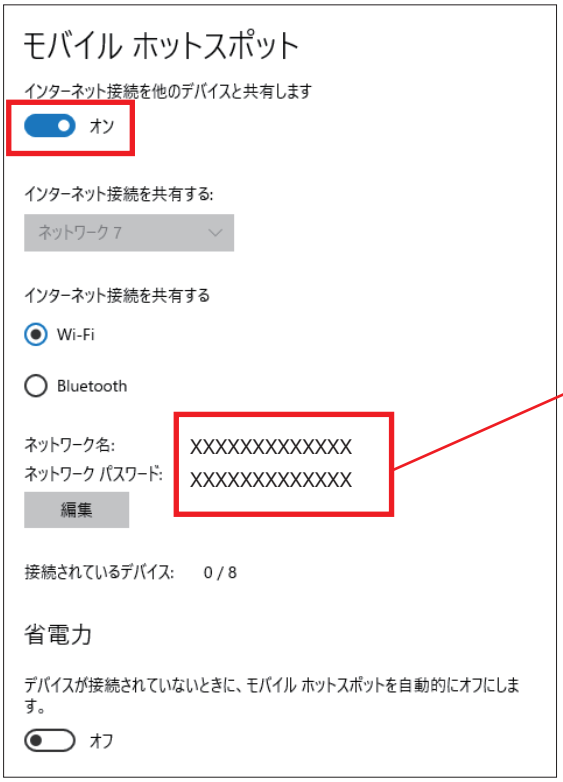


④複数のインターネット接続がある場合は、共有するインターネット接続を選択して、「Wi-Fi」を選択します。



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。



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

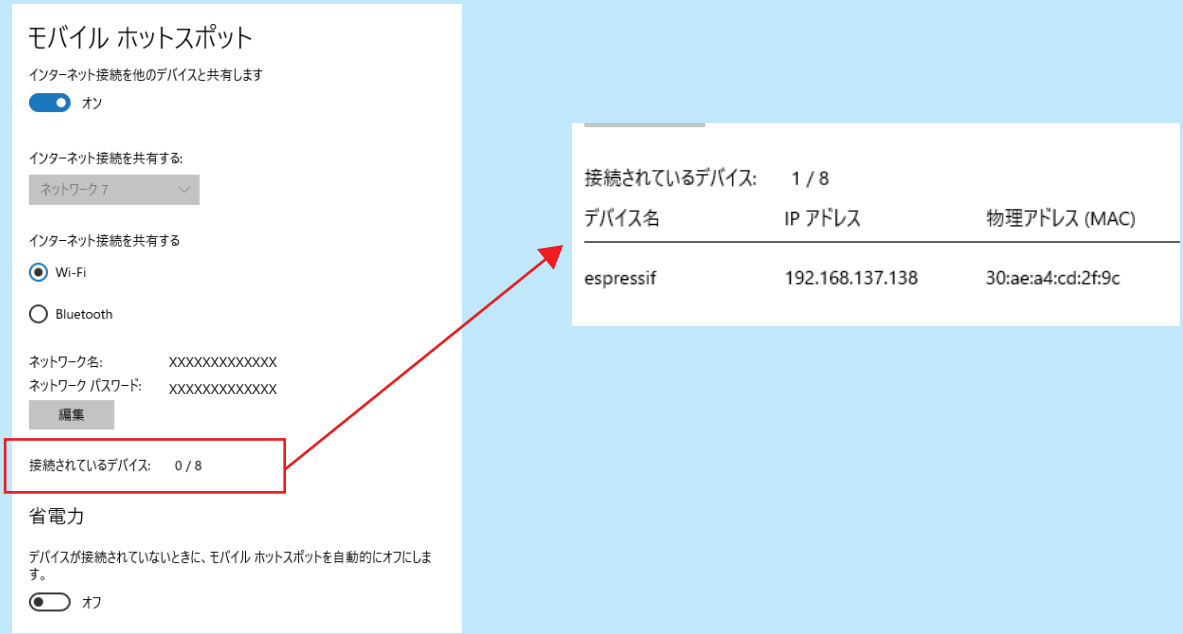
●モバイルホットスポットの設定画面に表示される IP アドレスについて

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



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. 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 G. cgi モジュールを使用しないプログラム

演習⑤、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⑤、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⑤ 演習⑤の device.py を下記の通り書き換えます。

※下記のos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。

Windows 環境

device.py

```
import os

print('Content-Type: text/plain¥n')
param = os.environ.get('QUERY_STRING')
dat = param.split('=')
val = dat[1]

if int(val) >= 30:
    cont = 'ON'
else:
    cont = 'OFF'
print(cont)
```

Studuino:bit から送られたパラメータを取得

受信したデータ‘sens= 温度’を‘=’で区切り、配列にして dat に代入する

dat=[sens, 温度] の「温度」を val に代入する

温度センサの値 (val) が 30℃以上の場合は ON、30℃未満の場合は OFF を cont に代入する

cont をレスポンスとして Studuino:bit に送信する

演習⑥ 演習⑥の index.py と device1.py を下記の通り書き換えます。

※下記の sys モジュール、os 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。

index.py

```
import sys,io

# ファイルにデータを書き込む関数
def fwrite(data, fn):
    file = open(fn, mode='w', encoding='utf-8')
    file.write(data)
    file.close()
# ファイルからデータを読み込む関数
def fread(fn):
    file = open(fn, mode='r', encoding='utf-8')
    text = file.readline()
    file.close()
    return text

# Web ブラウザに表示する HTML のコード
html = '''
<!DOCTYPE html>
<html lang="ja">
  <head>
    <meta charset="UTF-8">
    <title>IoT の実験 </title>
  </head>
  <body>
    <h2> 温度：{0}℃
      <input type="button" value=" 更新 "
       onclick="window.location.reload();">
      <form action="index.py" method="post">
        LED：
        <input type="submit" name="btn1" value="ON">
        <input type="submit" name="btn2" value="OFF">
      </form>
    </h2>
  </body>
</html>
'''
```

※次のページに続く

※前のページの続き

```
# utf-8 で標準出力を開き直すことで正しく utf-8 で出力されるようにする
sys.stdout = io.TextIOWrapper(sys.stdout.buffer, encoding='utf-8')
print('Content-Type: text/html; charset = utf-8¥n¥n')

# ファイル名
fsens = 'cgi-bin/sens.txt'  # Studuino:bit から受信したデータ
fcont = 'cgi-bin/cont.txt'  # ブラウザのボタンで設定した ON/OFF 情報
form = sys.stdin.readline()  # ブラウザのボタンの状態を取得する
if form != '':
    dat = form.split('=')  #'btn1=ON' または 'btn2=OFF' を '=' で区切る
    if dat[1] == 'ON':    # ON が押されている場合ファイルに ON を書き込む
        fwrite('ON', fcont)
    elif dat[1] == 'OFF':  # OFF が押されている場合ファイルに OFF を書き込む
        fwrite('OFF', fcont)
else:
    fwrite('OFF', fcont)
val = fread(fsens)  # Studuino:bit から受信したデータを読み込む
print(html.format(val))   # ブラウザに読み込んだデータを反映した HTML コードを表示
```

※「¥」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。

device1.py

```
import os

# ファイルにデータを書き込む関数
def fwrite(data, fn):
    file = open(fn, mode='w', encoding='utf-8')
    file.write(data)
    file.close()
# ファイルからデータを読み込む関数
def fread(fn):
    file = open(fn, mode='r', encoding='utf-8')
    text = file.readline()
    file.close()
    return text

print('Content-Type: text/plain¥n')
# ファイル名
fsens = 'cgi-bin/sens.txt'  # Studuino:bit から受信したデータ
fcont = 'cgi-bin/cont.txt'  # ブラウザのボタンで設定した ON/OFF 情報

param = os.environ.get('QUERY_STRING') # Studuino:bit から送られたパラメータを取得
dat = param.split('=') # 受信したデータ‘sens= 温度’を‘=’で区切り、配列にして dat に代入する
val = dat[1] #dat=[sens, 温度 ] の「温度」を val に代入する
if val != '':
    fwrite(val, fsens)    # sens の値をファイルに書き込む
cont = fread(fcont)     # ブラウザの ON/OFF 設定を読み込む
print(cont)     # ON/OFF 情報を Studuino:bit に送信
```

※「¥」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。