展開演習Bテキスト

椙山女学園大学現代マネジメント学部　三木　邦弘

令和5年9月27日版

1 QR コードの利用 1
- 1.1 PHP による QR コードの出力 1
- 1.2 QR コードによる決済 2
- 1.3 共通部分をまとめる方法 5
- 1.4 時刻の扱い方 6

2 ファイルのアップロードとデータベースへの格納 8
- 2.1 ファイルを送るための HTML のタグ . 8
- 2.2 ファイルを受け取るプログラム 9
- 2.3 ファイルをデータベースに入れる . . . 9
- 2.4 データベースから取り出したファイルの扱い方 10

3 物体検出プログラムの利用 11
- 3.1 PHP で Unix のコマンドを実行する . 11
- 3.2 PHP による python プログラムの実行 11
- 3.3 物体検出プログラムのオプション . . . 12
- 3.4 PHP による画像認識 12

4 メールと定期的処理 14
- 4.1 HTML でメールを送る 14
- 4.2 PHP でメールを送る方法 14
- 4.3 at を利用した予約実行 15
- 4.4 cron を利用した定期的実行 16

5 IoT とグラフの表示 18
- 5.1 どうやってつなぐか 18
- 5.2 電源をどうするか 19
- 5.3 センサーをどうするか 19
- 5.4 集めた情報の処理 19
- 5.5 環境センサーからの情報の取り込み . . 20
- 5.6 グラフの描画 21
- 5.7 グラフ化するデータの作成 25
- 5.8 超えたらメールを送る 27
- 5.9 web ページの自動更新 27

6 スクレイピング 28
- 6.1 Web ページの取り込み 28
- 6.2 HTML の解析 29

7 API の利用と XML・JSON 32
- 7.1 XML とは 32
- 7.2 JSON とは 32
- 7.3 API サービスの例 33
- 7.4 天気予報データの取得 35

8 ChatGPT 37
- 8.1 ゲームプログラムを作る 38
- 8.2 CharGPT の実行方法 45
- 8.3 ChatGPT をどのように捉えるか . . . 49

9 演習課題 51
- 9.1 QR コードの表示・QR コード決済システム (1) 9/27 51
- 9.2 QR 決済システム (2) 10/04 52
- 9.3 QR 決済システム (3) 10/11 53
- 9.4 QR 決済システム (4) 10/18 54
- 9.5 画像データベース 10/25 56
- 9.6 YOLO による物体検出 11/1 56
- 9.7 メールの定期送信 11/8 58
- 9.8 利用者登録の確認 11/15 58
- 9.9 円相場のグラフ 11/22 59
- 9.10 環境センサーのデータのグラフ化 11/29 60
- 9.11 スクレイピングと POST でデータを送る 12/6 61
- 9.12 天気予報の表示 12/13 61
- 9.13 シューティングゲームの作成 12/20 . . 62
- 9.14 志願理由書作成システム 1/10 62
- 9.15 API を利用した Chat システム 1/17 . 63

索引 66

1.　QR コードの利用

様々な商品に付けられているバーコードは、十数桁の数字を棒の太さや間隔で示したもので、通常会社コードと商品番号を合わせた数字を示しています。コンビニなどではこれをレーザーなどを利用して読み取り、データベースに照会して商品名や価格の情報を得て、レシートに印字します。バーコードの形と数字の内容が規格化されており、パッケージなどにバーコードを印刷するだけで安く使用できるのと、読み取り機が安価なため、広く使われています。ただバーコードで表現できる内容が数字で十数桁しかないために、個々の商品に異なる番号を振る事ができませんし、情報量が少なすぎるので商品番号以上の商品に関する情報を含めることもできません。QR コードは、1994 年 (平成 6 年) に自動車部品メーカーであるデンソーの開発部門が開発したマトリックス型二次元コードです。デンソーは QR コードを広く普及させることを考えて、特許をオープンにしました。最大約 7 千桁の数字が表現できる他、用途に合わせて誤り訂正レベルが設定でき、スマホなどで読み取り可能なため、現在 QR 決済などで様々な分野で広く使われています。

この章ではこの QR コードを扱う方法の説明や、簡単な QR 決済のシステムを作成します。

1.1　PHP による QR コードの出力

Web ページに QR コードを表示させる場合、いつも同じ内容の QR コードであれば、QR コードの画像ファイルを用意して、HTML の Img タグでその画像ファイルを指定すれば可能です。一方お客さんにお互いに異なる座席を指定する電子チケットのような QR コードは、全て異なるものでないと困ります。そしてそれを予め全て画像として用意しておくより、必要に応じて作成する方がよいでしょう。

画像を作成できるプログラムが書けるのであれば、基本的に QR コードの画像を生成するプログラムを書くことは可能です。しかしそのプログラムを書くためには QR コードの細かい仕組みを理解する必要があり大変です。幸いなことに様々なプログラミング言語用の QR コード生成プログラムが公開されているので、それを利用させていただくことにします。基礎演習のテキストの AR システムでは JavaScript によるものを使用していますが、PHP 用の QR コードを生成するプログラムの一つをここでは紹介します。

プログラムの作者は Y.Swetake 氏で https://www.swetake.com/で公開されています。このサイトからダウンロードしたものはまとめて圧縮されているので、解凍・展開をします。出てきたファイルの中で必要なものは、php ディレクトリの中の `qr_img.php` と data ディレクトリと image ディレクトリです。この 3 つを QR コードを表示する HTML や PHP のファイルがあるところにコピーし、`qr_img.php` の最初の方にある以下の部分を書き換えます。

```
$path="./../data";           /* You must set path to data files. */
$image_path="./../image";    /* You must set path to QRcode frame images. */
        ↓
$path="data";           /* You must set path to data files. */
$image_path="image";    /* You must set path to QRcode frame images. */
```

そして QR コードを表示したいところに、以下のように HTML の Img タグを記述します。

```
<Img src="qr_img.php?d=http://www.sugiyama-u.ac.jp/&e=L&s=3">
```

- d=で QR コードで表したいデータを指定します。例のように URL でも良いし、単なる数値などでも構いません。漢字は読み取り側がどのような漢字コードと解釈するのか明確でないので避けましょう。URL のファイル名の後に?が付いてパラメタがある場合は、e=以下の指定と混ざってしまうので、urlencode() を使用して変換したものを指定します。

- e=で誤り訂正のレベルを指定します。L、M、Q、H のどれかを指定します。L が一番誤り訂正レベルが低く、H が一番誤り訂正レベルが高くなります。訂正レベルが高いほど QR コードの汚れなどに強くなりますが、図 1.1 を見ても分かるように、QR コードが大きくなります。何も指定しない場合は e=M を指定したことになります。
- s=で QR コードの大きさを指定します。1 の場合が一番小さくなります。何も指定しない場合は s=4 を指定したことになります。

パラメタを変えると図 1.1 のように QR コードを表示することができます。QR コードで表しているのは全て同じ「https://www.sugiyama-u.ac.jp/」です。

図 1.1　QR コード出力の例

1.2　QR コードによる決済

図 1.2 のように我々がお店で何か買い物をする際には、代金を現金で支払うのが普通でした。支払いが高額な場合、現金を持ち歩くのは危ないので、クレジットカードによって支払うという方法が考えられました。クレジットカードを見せて、サインすると支払いの手続きは終わります。お店はクレジットカードの番号を元にクレジット会社に請求し代金を得ます。クレジット会社は銀行の口座からお店に渡した代金を引き落とします。銀行から代金を引き落とすためには、手数料が必要になります。その手数料は、通常お店から取った代金の数 % 分になるクレジットカードの利用料から支払います。お店の売上が利用料の分減りますので、クレジットカードの使用ができない店や、クレジットカードの支払いの際には割引ができない店があります。

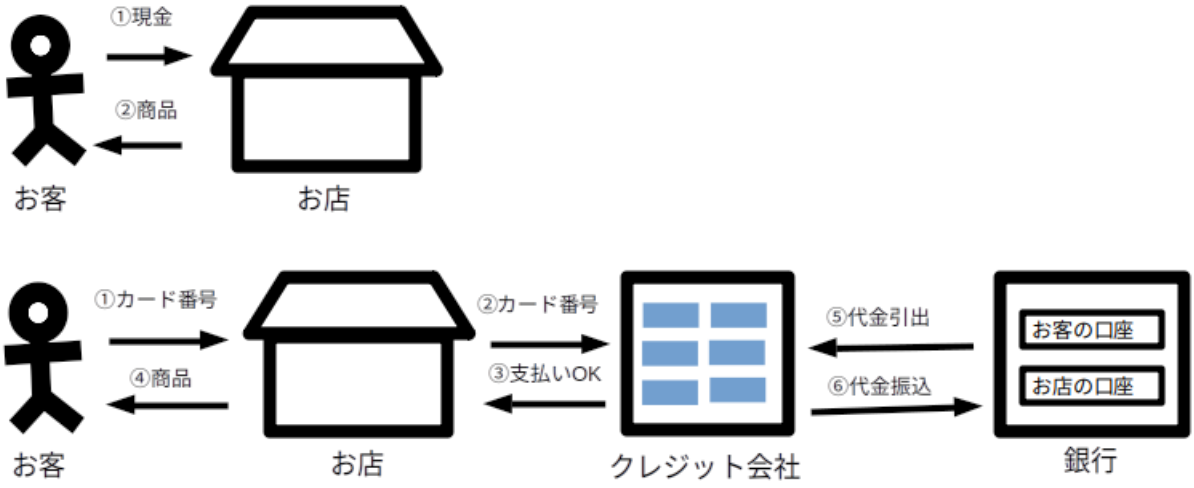


図 1.2　現金やクレジットカードによる支払い

スマホの決済や QR コードによる決済も基本的にはクレジットカードと同じです。QR コードによる決済の利点は、お店側に高価な設備が必要でないところでしょう。スマホで済ませることも可能です。お店にとっての問題はクレジットカードと同様に利用料を取られるところです。銀行の手数料がある限り、代金の大きさに関わりなくある程度の手数料を取られます。この問題を解決するのに一番簡単な方法は、お店や利用者の口座を、図 1.3 のように QR コードの決済システムの中に持つことです。同じシステムの中で数字が動くだけですので、コストはほぼゼロになります。現在の日本ではいくつかの決済システムが覇権を争っているところです。大きくなるほどシステム内でのお金のやり取りになるのでコストが減ります。赤字覚悟で利用者に還元しても将来取り返せるということでしょう。他社が無くなっていれば、取り返すために手数料を上げても逃げられません。

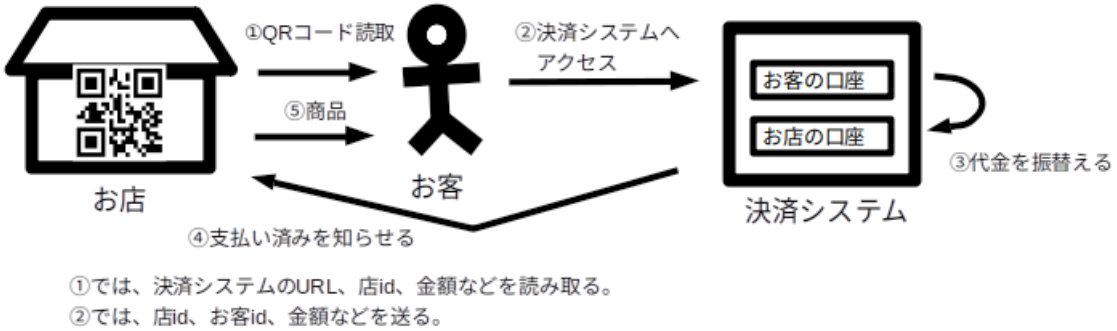


図 1.3　QR 決済による支払い

クレジットカードの利用はある程度高額な支払いでしたが、QR コードによる決済などではより少額の支払いにも使われます。誰がいつ何を購入したかがわかると、売る側は次に何が売れるか予想を立てることができます。そういう情報の収集にも役立つと考えているようです。

ここでは次のように使う QR コード決済システムを作ってみましょう。QR コードには、決済システムの入り口の URL のみ、URL と支払い先の ID、URL と支払先の ID と支払う金額の 3 つのパターンがあるものとします。

1. お客やお店の人が決済システムにアクセスすると、既に認証済みであれば、入出金履歴リストが表示されます。もし認証がまだであれば、名前とパスワードを入力して認証を通過すると同じものが表示されます。
2. お客がお店の人の ID 付きの QR コードを元にアクセスすると、既に認証済みであれば、お客の残額が表示された支払額の入力画面が表示されます。もし認証がまだであれば、名前とパスワードを入力して認証を通過すると同じものが表示されます。支払額を入力し送信すると支払いが終了します。お店の人は入出金リストを更新し、支払いがあったことを確認してから商品をお客に渡します。
3. お客がお店の人の ID と支払額付きの QR コードを元にアクセスすると、既に認証済みであれば即座に支払いが終了します。もし認証がまだであれば、名前とパスワードを入力して認証を通過すると支払いが終了します。

これまでの話に出てこなかった認証が出てきます。大事なお金のやり取りですから、認証を使って第三者が勝手に触れないようにする必要があります。一方お買い物の度に認証をするのは面倒です。そこで前期にやった Cookie などを利用して一度認証を通過すれば、しばらくは名前とパスワードを入れなくて済むようにします。一方そのしばらくの間に他の人に使われる恐れが生じます。心配性の方のために認証済みを解除できるようにする必要もあります。

お客が支払額を入れ間違えると面倒なことになります。足りない場合は足りない分を再度支払えば良いので

すが、多すぎた場合は返金の仕組みが必要になります。返金はお客の QR コードをお店の人が読み取れば可能ですが面倒です。そこで QR コードに金額の情報も付けられるようにして、お客が支払額を入力する必要がないようにしています。ただこの場合、支払金額によって QR コードが変わります。よってお店の QR コードを印刷して使うという方法は無理で、レジに QR コード表示用の画面が付いていて、そこに支払額込みの QR コードを出すようなシステムが必要になります。

これらを実現するために図 1.4 のような構成にします。

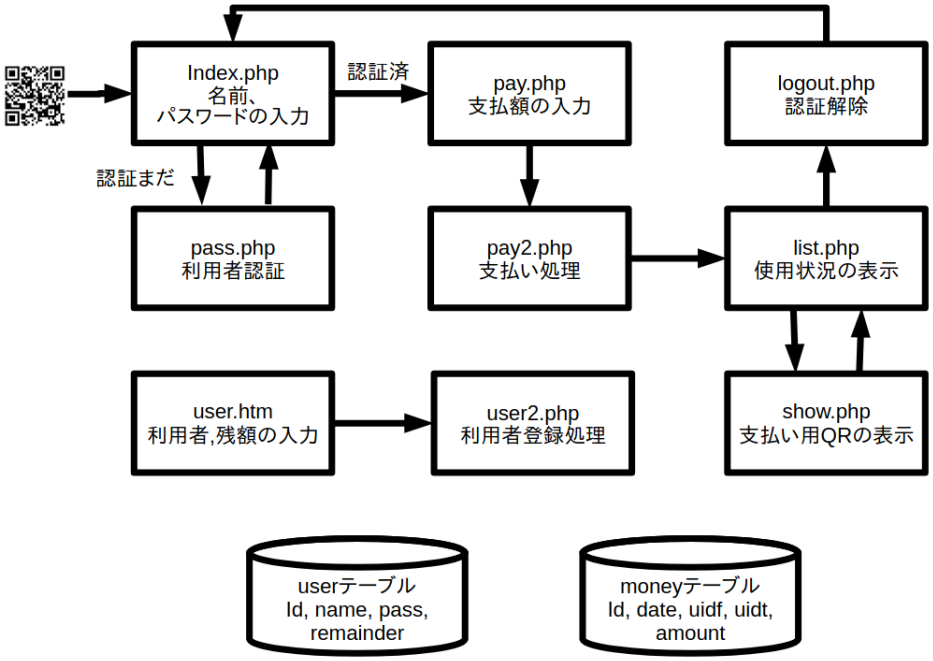


図 1.4　QR コード決済システムの構成

- user テーブルで、名前 (name)、パスワード (pass)、残額 (remainder) を管理します。
- money テーブルで、いつ (date)、誰 (uidf) が、誰 (uidt) に、いくら (amount) 送金したか記録します。
- user.htm で入力した内容を user2.php で user テーブルに登録することにより利用者登録をします。本来ならばきちんと管理者かどうかの認証をする必要がありますが、ここでは省略しています。
- お客やお店の人が index.php にアクセスした場合、もし URL に支払先や金額の情報があれば session 変数に入れます。そして認証済みかどうか調べます。認証がまだの場合はここで名前とパスワードを入力して、pass.php で user テーブルの内容と合うかどうかで判定します。認証済であれば、pay.php へ行きます。
- pay.php では、session 変数を調べて支払金額が未定であればお客に残額を表示し、支払額を入力してもらい pay2.php へ行きます。もし session 変数に支払金額があれば即座に pay2.php へ行きます。
- pay2.php では、お客からお店へ支払額が動いたと言うデータを money テーブルに追加し、user テーブルの残額も修正します。そして list.php へ行きます。
- list.php では入出金リスト、自分宛ての支払い用の QR コードを表示する show.php へのリンク、認証を解除する logout.php へのリンクを表示します。
- お客の支払い処理が終わったら、お店の人は list.php を更新し、支払いがあったことを確認します。
- logout.php では、認証済み情報を削除します。これによって共有の端末で使用した場合に、他の人に使われてしまうことを防ぎます。
- 図 1.4 にはありませんが、データベースを利用する際に必要になる部分を common.php に入れて共有

します。

1.3 共通部分をまとめる方法

複数のファイルで構成されたシステムでは、ファイルの内容に共通な部分がよく見られます。例えばデータベースを利用する際のデータベースを開いたり、トランザクションをスタートする部分です。このような共通する部分は、別のファイルに入れておいてそれを取り込む形にすると、各ファイルでは別ファイルを取り込む指示を入れるだけで済みます。またこの共通部分に修正が生じた場合も、別ファイルを修正するだけで済みます。共通するファイルの書き方は次のようになります。

```
<?php
共通する部分
?>
```

これが「common.php」と言う名前のファイルに入っている場合、この内容を取り込みたいところに、次のような指示を追加します。

```
include "common.php";
```

もちろんこれは PHP の機能ですので、<?php〜?>の間になければいけません。

共通する部分が、一つのファイルの中に何回か出てくる場合があります。このような場合は関数を利用します。PHP の関数は JavaScript の関数と同じ形をしています。例えば次のような形で関数の定義をします。

```
function iloveyou() {
   for ($i=0;$i<100;$i++) {
      echo "I love you. ";
   }
   echo "<Br>";
}
```

関数の定義だけでは、関数の内容が記憶されるだけで「I love you.」は出てきません。そして出したいところに、次のように記述すると「I love you.」が 100 回表示されます。

```
iloveyou();
```

いつも 100 回ではなく、違う回数の場合もあると言う場合は、関数に注文を付けることが可能です。その場合は次のように関数を定義します。

```
function iloveyou($kai) {
   for ($i=0;$i<$kai;$i++) {
      echo "I love you. ";
   }
   echo "<Br>";
}
```

最初の行の () の間に適当な変数を書きます。これが注文を受け付ける変数です。そして次のように関数を呼びます。

```
iloveyou(100);
iloveyou(10000);
```

1 つ目では 100 回、2 つ目では 10000 回の「I love you.」が表示されます。複数の注文を受ける形の関数も可能です。回数だけでなく、love する相手も変えたい場合は次のようにします。

```
function iloveyou($kai,$aite) {
   for ($i=0;$i<$kai;$i++) {
      echo "I love ",$aite,". ";
   }
   echo "<Br>";
}
```

注文を受ける変数を「,」で区切って並べます。呼び出す方も同様に「,」で区切ります。

```
iloveyou(3,"Miki sensei");
iloveyou(1000,"cats");
```

関数の中の変数は関数の中のみで有効です。呼び出す側と同じ名前の変数を関数の中で使っていても、別物として扱われます。また関数の実行が終わったら関数の中の変数は消えてしまいます。呼び出す側と同じ変数を関数の中で利用したい場合は、次のようにします。

```
function showmsg() {
   global $msg;

   echo $msg,"<Br>";
}
```

「global」の後に書いた変数は、関数の外のものと共通になります。よって showmsg() を呼ぶ前に、$msg になにか入れておく必要があります。

1.4　時刻の扱い方

コンピュータのハードウェアでは時刻は「2023 年 10 月 14 日 16 時 45 分 20 秒」と言うような形で扱っていることが多いと思います。水晶振動子による正確な周波数の信号を数える IC があるからです。一方 Windows や Unix のような OS のレベルでは、ある時点からの秒数などで管理しています。我々は現在時刻をある時点からの秒数で言われても使えませんが、期間の計算やどちらが先かなどの判定が簡単にできるからです。PHP では Unix と同じく時刻を、グリニッジ標準時の 1970 年 1 月 1 日の 0 時からの秒数で管理しています。

PHP には現在の時刻を求めるために time() と言う関数があります。この関数は呼び出された時刻を示す秒数を返します。内部ではこの数値のまま扱う方が便利です。例えばちょうど 24 時間後の時刻を求めたいときは「24*60*60」を加えます。これが通常の「2023 年 9 月 30 日 16 時 45 分 20 秒」のような形では、24 時間後に月や年が変わる場合があるので面倒です。データベースに記録するような場合も大抵秒数のまま入れます。

時刻を表示する場合は秒数ではいつなのか分からないので、PHP には秒数を指定した形式の日時の形に変える date() と言う関数があります。形式の指定方法が少し面倒ですが、曜日だけ出すようなこともできます。

```
echo time(),"<Br>";
echo date("Y/m/d H:i:s"),"<Br>";
echo date("Y/m/d H:i:s",time()+12*60*60),"<Br>";
```

これを実行すると、例えば次のようなものが表示されます。2 行目のように、date() で 2 つ目の秒数の指定がない場合は、現在の時刻が表示されます。

```
1691659440
2023/08/10 18:24:00
2023/08/11 06:24:00
```

date() で日時の形式指定は表 1.1 のような文字を使います。それ以外の空白や「/」のような記号はそのまま出てきます。

表 1.1 日時の形式指定文字 (一部)

文字	意味	文字	意味
s	秒 (00 〜 59)	D	曜日 (Mon 〜 Sun)
i	分 (00 〜 59)	w	曜日 (0 (日曜) 〜 6 (土曜))
g	時 (1 〜 12)	M	月 (Jan 〜 Dec)
h	時 (01 〜 12)	m	月 (01 〜 12)
G	時 (0 〜 23)	n	月 (1 〜 12)
H	時 (00 〜 23)	Y	年 (1970 〜)
d	日 (01 〜 31)	y	年 (00 〜 99)
j	日 (1 〜 31)		

逆に mktime() を利用すると通算秒数を求めることができます。

```
echo mktime(12,34,56,9,27,2023);
```

これで「1695785696」と言う数字が表示されますが、これは 2023 年 9 月 27 日 12 時 34 分 56 秒の通算秒数です。年月日の指定の順番がアメリカ風？に時、分、秒、月、日、年の順番になっているのでご注意ください。

2.　ファイルのアップロードとデータベースへの格納

ファイルを Web サーバーに送ることができます。利用者から送られたファイルの内容をファイルの形で保存したり、データベースに入れる方法についてここでは説明します。従来はパソコンやスマホに存在するファイルを送ることしかできませんでしたが、HTML の ver. 5 では、その場で撮影した画像や動画を送ることもできるようになりました。なお Web サーバーの設定でアップロードできるファイルの大きさには制限がされていますので、巨大な動画ファイルなどはアップロードできません[*1]。

2.1　ファイルを送るための HTML のタグ

ファイルを Web サーバーに送る際には、これまでと少し形の異なる Form タグを使用します。

```
<Form method="POST" enctype="multipart/form-data" action="upload.php" name="aaa">
```

「method=」は必ず「POST」にしてください。「enctype=」の指定も必ず追加します。「name=」の指定は必須ではありませんが、サーバーに送るファイルをドラッグ＆ドロップで即座に送りたいと言う場合は、JavaScript の助けが必要となるので名前を付けておく必要があります。アップロードするファイルを指定するボタンは次のタグで出すことができます。

```
<Input type="file" name="ufile">
```

これで通常参照...と言うようなボタンが表示され、クリックするとファイルが選択できるようになります。またこのボタンの上にドラッグ＆ドロップでアップロードするファイルを指定することもできます。

HTML ver. 5 より「capture=」と「accept=」の指定をすると、スマートフォンでの使用に限定されますが、その場での録音や撮影した結果をアップロードできるようになりました。ただしパソコンでアクセスした場合は、通常のファイルのアップロードと同じ扱いになります。次のようにすると内側のカメラで撮影した画像をアップロードできるようになります。

```
<Input type="file" name="cfile" capture="user" accept="image/*">
```

「capture=」を「environment」にすると外側のカメラになります。また「capture=」の指定を省略するとスマートフォン側でカメラやファイルを選択できるようになります。「accept=」を「audio/*」にすると音声を、「video/*」にすると動画をアップロードするようになります。

ドラッグ＆ドロップで即座に送りたいと言う場合は次のようにします。

```
<Input type="file" name="dfile" accept="image/*"
       style="width: 20em; height: 5em; display: block; border: 2px solid red"
       onDrop="setTimeout('document.aaa.submit()',500)" >
```

ドロップしやすいようにスタイルの指定で、大きなボタンにしています。また「onDrop=」で 0.5 秒後に送信するようにしています。このように少し待ってから送信しないと、ファイルの内容がうまく送れないことがあります。

「onDrop=」の指定がない場合は従来と同じように送信ボタンが必要です。

```
<Input type="submit" value="アップロード">
```

[*1] mars の現在の設定では、同時に 20 個のファイル送ることができ、合計のサイズは 16MB になっています。

2.2 ファイルを受け取るプログラム

「ufile」と言う名前の入力欄から送られてきたファイルに関しては次のような変数が使用可能になります。

変数	変数の内容
`$_FILES['ufile']['tmp_name']`	アップロードされたファイルのとりあえずの名前
`$_FILES['ufile']['name']`	アップロードされたファイルの元々の名前
`$_FILES['ufile']['size']`	アップロードされたファイルの大きさ
`$_FILES['ufile']['type']`	アップロードされたファイルの MIME タイプ

これらといくつかの関連する関数を使用して、PHP で送られてきたファイルの内容を保存する例を以下に示します。

```
if (is_uploaded_file($_FILES['ufile']['tmp_name']) &&
    $_FILES['ufile']['size']>0) {
   move_uploaded_file($_FILES['ufile']['tmp_name'],"aaa.jpg");
}
```

ファイルを選択せずに送信することがありますので、`is_uploaded_file()` を使用してファイルがアップロードされたか確認をします。さらに送られたファイルの大きさが空の場合も外していますが、ここの条件は状況に合わせて変更したり省略することも可能です。`move_uploaded_file()` でとりあえずの名前のファイルを「aaa.jpg」に変更しています。とりあえずの名前のファイルは PHP の実行が終わると消去されますので、必ず `move_uploaded_file()` をします。またこの例のように固定の名前 (aaa.jpg) では、アップロードしたファイルはどんどん上書きされるので、上書きされては困る場合は、毎回異なるファイル名を指定する必要があります。

2.3 ファイルをデータベースに入れる

かつてのデータベースは、数字、文字列などの小さなサイズのデータを大量に扱うものでしたが、現在では画像などの大きなサイズのデータも扱うことが可能です。これによって例えば写真画像のデータベースで、写真画像と共に撮影者、撮影日時、撮影対象などの情報をまとめて扱うことができます。SQLite では、データ型として整数値を入れる「INTEGER」文字列を入れる「TEXT」に加えて、バイナリーデータを入れる「BLOB」が使用可能です。ただバイナリーデータはその中に区切りの「'」などを含むこともあるため、これまで説明したのと少し異なる方法を使います。

次の例は「photo] テーブルの TEXT 型の「name」と BLOB 型の「image」にそれぞれ「sample」と「star.png」ファイルを挿入するものです。$sql に prepare() でまず SQL 文の雛形を入れて、bindParam() で「:image」の部分に入れる内容を指定しています。

```
$sql=$db->prepare("INSERT INTO photo (name, image) VALUES ('sample', :image )");
$img=file_get_contents("star.png");
$sql->bindParam(':image',$img,PDO::PARAM_LOB);
if (!($sql->execute())) { die("Insert できません "); }
```

なお、違う内容のデータを挿入したい場合は、bindParam() と execute() をやり直すだけでできます。そのような場合は、name の方も「:」で始まるパラメタにして、bindParam() も 2 個にします。

データベースの内容を修正する場合 (UPDATE) も、BOLB 型のデータは上記の例と同様に prepare() を使います。一方データを取り出す (SELECT) などは、これまでと同じやり方で変数に取り出すことができます。ただ数値や文字列と異なり画像や動画などは echo で表示することはできません。

2.4 データベースから取り出したファイルの扱い方

データベースから取り出したファイルをブラウザに出すには、ファイルの内容を元に、ファイルを作成するのが一つの方法です。`$data['image']` と言う変数にファイルの内容が入っている際に、次の 2 行でファイル (test.png) を作成することができます。

```
file_put_contents("test.png",$data['image']);
chmod("test.png",0644);
```

ファイルの作成は `file_put_contents()` で終わっていますが、chmod() でファイルの読み書きの設定を変更しないと、外部からアクセスができません。「0644」で自分はファイルの読み書きできるが、それ以外のものにはファイルの読み出ししか許さないようになります。できた画像ファイルを HTML の Img タグで指定すれば表示されます。A タグで指定すればリンクをクリックすると単独で表示されます。動画ファイルであれば Video タグ、音声ファイルであれば Audio タグで指定すれば視聴できるようになります。PDF ファイルであれば A タグか iFrame タグが良いでしょう。

このファイルを作成するやり方にはいくつかの問題点があります。複数の画像を表示する場合、別のファイル名にするのが無難です。同じファイル名を使い回すと、内容が変わってもブラウザ側にそれが伝わらず、以前の画像のまま変わらないようなことが生じます。またこうして作成したファイルはいつまで置いておくのでしょうか。ほっておくとデータベースにある全てのファイルがでてきてしまいます。

もう一つのやり方は、必要に応じて画像ファイルなどの内容を送るという方法です。展開演習で QR コードを表示する方法についてやりましたが、QR コードは次のようにすれば表示されるとしていました。

```
<Img src="qr_img.php?d=http://www.sugiyama-u.ac.jp/&e=L&s=3">
```

これは「`qr_img.php`」が「?」の後に並ぶパラメタを元に QR コードの画像を生成し、ブラウザに送っています。これと同じことをすれば、異なるファイル名を考える必要もなく、作ったファイルの消去のタイミングも考えなくて済みます。ただ毎回画像の内容を送るために、画像の一覧のようなページの表示に毎回時間がかかることがあります。

それでは「`qr_img.php`」ではどのようなことを行っているのか。パラメタからデータベースを検索し、既に表示すべき内容が`$data['image']` と言う変数に入っているとします。

```
header('Content-Type: image/png');
echo $data['image'];
```

header() を使用しますので、HTML のタグなどを出さないようにします。「Content-Type:」で指定する MIME タイプはファイルの種類によって変更する必要があります。次の表のようにしてください。

ファイルの種類	指定するもの	ファイルの種類	指定するもの
画像 (PNG)	image/png	画像 (JPEG)	image/jpeg
音声 (MP3)	audio/mpeg	動画 (MP4)	video/mp4
文書 (PDF)	application/pdf		

header() と echo の間に次のような header() を追加することにより、ブラウザ側で表示するのではなく、ダウンロードするように指定することもできます。

```
header('Content-Disposition: attachment; filename="star.png"');
```

3. 物体検出プログラムの利用

ここでは基礎演習で紹介した YOLO ver.5[*2]を使用して物体を検出する方法を説明します。基礎演習で作成した「yolov5」ディレクトリを「www」ディレクトリの中に移動して使います。基礎演習の際は端末ソフトを起動して、python のコマンドを入力して動かしましたが、PHP を利用して YOLO を動かして、ブラウザで結果が見えるようにします。

3.1 PHP で Unix のコマンドを実行する

Unix には python 以外にも多くのコマンド (実行プログラム) があります。昔々文字入力と文字出力しかできなかった時代に誕生したコマンドは、文字で制御できるので、簡単に他のプログラムから利用することができます。ここでは PHP から Unix のコマンドを利用する方法を説明します。まずコマンドを実行するだけならば次のようになります。

```
system("date");
```

これで Unix の date コマンドが実行されて、その出力がブラウザに送られます。echo は必要ありません。ブラウザには「Thu Oct 10 04:26:26 PM JST 2023」と言うような実行された日時が表示されます。パラメタが必要な場合は、空白を空けてコマンドの後に入れます。date は世界標準時で表示する際は「-u」を指定することになっているので次のようにします。

```
system("date -u");
```

この場合ブラウザには「Thu 10 Oct 2023 07:26:26 AM UTC」のように表示されます。

コマンドを起動してからさらに入力が必要な場合は次のようにします。

```
$p=popen("Unix のコマンド","w");
fwrite($p,"コマンドに送る内容\n");
pclose($p);
```

popen() がコマンドを起動し、”w”の指定によりコマンドに送ることができるようになります。fwrite() を必要な数だけ使い、最後に pclose() でコマンドに、これで終わりであることを伝えます。

popen() で”w”の代わりに”r”を指定すると、コマンドの出力をファイルから読み込むのと同様に fgets() で取り込むこともできます。

3.2 PHP による python プログラムの実行

基礎演習では次のようなコマンを入力してサンプル画像の実行を行いました。

```
python detect.py --source data/images/bus.jpg
```

これを PHP で実行させるために前章で説明した system() を使うとすると次のようになります。

```
system("python detect.py --source data/images/bus.jpg");
```

[*2] 2020 年 6 月に Glenn Jocher が発表した YOLO ver.5 (https://github.com/ultralytics/yolov5)

残念ながらこれは大抵の場合エラーが表示されてうまく動きません。エラーは detect.py が見つからないと言うものです。これはディレクトリの指定が適切に行われていないからです。例えば「QR」ディレクトリーの中にある PHP のプログラムで単に「data.db」ファイルを指定すると、同じ「QR」ディレクトリの中にある「data.db」ファイルと解釈されます。大抵の場合、PHP のプログラムと detect.py は別のディレクトリに居るので、見つからないと言うエラーになります。この問題を解決するには、detect.py の指定の際にディレクトリの情報を付ける方法がありますが、この場合は detect.py の中でまた別のファイルを参照しているとそこでエラーになります。Unix の「cd」コマンドを利用してディレクトリの移動を行う方が確実でしょう。「yolov5」ディレクトリのとなりにあるディレクトリにある PHP のプログラムで実行させる場合は、次のようにします。

```
system("cd yolov5; python detect.py --source data/images/bus.jpg");
```

「cd yolo5;」で今居るディレクトリから「yolov5」に入ります。

3.3　物体検出プログラムのオプション

YOLO の物体認識を行う detect.py は、特に指定がなければ毎回新しいディレクトリに結果を保存します。一番新しいディレクトリを探す方法では、複数の実行が続いた際に先に実行した結果でなく、後からの実行結果が入ったディレクトリを開いてしまう可能性があります。`--exist-ok` を付けて実行すると同じディレクトリを再使用するようになるので、いつも同じディレクトリに結果が入るようになります。

これまでの YOLO の実行例では物体認識結果を上書きした画像のみが得られていましたが、認識した結果を元にさらに処理を行う場合は、どこに何があったかが必要になります。`--save-txt` オプションを付けると別のテキストファイルに認識結果が入るようになります。さらに信頼度も必要な場合は、`--save-conf` オプションも追加します。

以上をまとめると次のようになります。

```
system("cd yolov5; python detect.py --source $fname$ext --exist-ok".
       " --save-txt --save-conf  2>&1");
```

一番最後に「`2>&1`」と言うおまじないのようなものが付いていますが、これを省略すると detect.py 実行中のメッセージがブラウザに送られないようになります。`--exist-ok` オプションのため認識結果の画像は、「yolov5」ディレクトリの中の「runs」の中の「detect」の中の「exp」の中にあります。

3.4　PHP による画像認識

ここでは PHP で python のプログラムを実行することにより、アップロードした画像にある物体を YOLOv5 で認識して表示するものを考えます。これまで他の章で取り上げた内容を組み合わせて実現します。

3.4.1　アップロードしたファイルの名前

画像ファイルのアップロードについては、既に 2 章 (p.8〜) でやっています。ここで問題になるのはアップロードしたファイルの名前です。いつも同じ名前のファイルにすると複数の利用者が違う画像ファイルをアップロードした場合、先にアップロードしたものが消えてしまいます。また通常 YOLOv5 で認識した結果のファイル名も同じものになりますが、同じ人が異なるファイルを何度も送った場合、結果のファイル名が同じ場合、ブラウザが違う内容のファイルだと認識してくれないので、再読込をしても最初と同じものが表示されてしまいます。

アップロードしたファイルの名前としては、ランダムなものを使用するのが良いのですが、次善の策としてアップロードした日時によるファイル名にすることが考えられます。time() を使って秒単位の名前にすればま

ず重なって消えることはないでしょう。異なる名前を使用した場合の問題点はどんどんファイルが貯まることです。定期的に古いファイルを削除するようにするか、アップロードした際に古いファイルがあれば一つ削除すれば良いでしょう。

元のファイルの拡張子は維持する必要があります。YOLO は PNG 形式でも JPEG 形式でも対応してくれるようですが、拡張子が違っているとだめなようです。文字列関数の strpos() で「.」を探して substr() で切り出すのが良いでしょう。ファイル名の途中に「.」があるような困ったファイル名が心配であれば、使い方は strpos() と同じですが、後ろから探してくれる strrpos() を使いましょう。

3.4.2 YOLO の実行結果

YOLO の実行結果の表示もどこに結果があるのかちゃんと指定する必要があります。アップロードしたファイルは yolov5 ディレクトリに入り、そのファイル名は`$fname` に、拡張子の部分 (例えば「.png」) は`$ext` に入っているものとします。PHP のファイルは「yolov5」ディレクトリと同じところにあるとします。次のようにして検出した結果を表示することができます。

```
<Img src="yolov5/runs/detect/exp/<?= $fname.$ext ?>">
```

また画像のあるところにある「labels」ディレクトリの中に認識結果の数字の入ったテキストファイル (拡張子は.txt) があります。その内容は認識した物体の数と同じ行数になっており、各行は 6 つの数字が空白で区切られています。

```
23 0.103333 0.621667 0.111667 0.171667 0.303357
16 0.1075 0.62125 0.121667 0.170833 0.366978
26 0.915417 0.617917 0.0791667 0.1175 0.37679
```

左から認識した物体のクラス番号、中心 X 座標、中心 Y 座標、幅、高さ、信頼度になっており、クラス番号は 0 以上の整数値、次の 4 つの数字は画像全体を 1 とした相対的な数値、最後の信頼度は 0〜1 の数値になっています。`--exist-ok` を付けて実行すると、認識結果がどんどん追加されるので、同じ画像ファイル名で実行する場合は、先にファイルを削除する方が良いでしょう。

クラスの番号が何を示すかは、学習の際に使用した拡張子.yaml のファイルの内容で分かります。YOLOv5 に付いている学習済みデータについては、「yolov5」ディレクトリの中の「data」の中にある coco128.yaml の内容を見れば分かります。

空白や「,」で区切られたデータから個々のデータを切り出す際には、explode() を利用することができます。今「23 0.103333 0.621667 0.111667 0.171667 0.303357」が$line に入っているとき、次のようにすると $data[0] に最初の空白までのもの (23) が、$data[2] に次の空白までもの (0.103333) が入ります。6 つのデータが 5 つの空白で区切られている場合は、このように$data[0]〜$data[5] に順番に入ります。

```
$data=explode(" ",$line);
```

4. メールと定期的処理

この章ではメールを送る方法と、定期的に処理を行う方法について説明します。メールはシステム側から何か伝えたい時に利用するのに便利なものです。web ページに大事なお知らせを掲載しても、利用者がアクセスしてくれないと伝わりません。その点メールであれば、多くの人がスマホのメールを利用しているので、即座に伝わります。LINE の方が確実という人も多いでしょうが、比較的閉じた利用者間で使う LINE を登録するのは抵抗がある人も多いと思います。

またお知らせとなると、定期的に知らせる形もあるでしょう。それほど急がない知らせをどんどんメールで送るより、決まった間隔でまとめたお知らせメールを送るという方が望ましいでしょう。使う時にしかスイッチが入らないパソコンでは、定期的な処理は難しい場合が多々あります。一方昔から 24 時間動かしているサーバーで用いられてきた Unix では、決められた間隔でプログラムを起動するような機能があります。ここではその使い方も説明します。

4.1 HTML でメールを送る

web ページでは URL がわかればアクセスできるのと同様に、メールはメールアドレスがわかればメールを送ることができます。web ページにおいて、問い合わせをメールで送ってほしい場合、メールアドレスを書いておくだけではそれを送り手は入力する必要が生じます。コピペで済ます人も居るでしょうが、操作ミスでメールアドレスが正しく貼り付けられないと届きません。このような場合、既に学んだ HTML の A タグでメールを送るリンクを作ることができます。

```
<A href="mailto:aaa@bbb.ccc">問い合わせ</A>
```

これで「問い合わせ」と言うリンクが表示されて、これをクリックするとブラウザに設定されているメーラー (メールを送るプログラム) が起動されます。そして送り先として「aaa@bbb.ccc」が自動的にセットされます。メールを送り損なう最大の原因であるメールアドレスの入れ間違いが生じないので、確実にメールをもらえる事になります。

この方法で問題になるのは、スマホでは余りないようですが、パソコンなどではブラウザにメーラーが設定されていない事が多い点です。そのために web ページの見えるところにメールアドレスも明記しておく事が考えられます。またメールアドレスがばれるので、嫌がらせなどの標的に利用される恐れもあります。

問い合わせの受付ならば、適当な入力欄を作成し、そこに問い合わせ内容を書いてもらい、「送信」ボタンをクリックしたら、PHP で入力欄の内容を取り出してファイルに入れれば良いでしょう。この場合の問題点は、問い合わせの答えをどのように返すかです。よくあるのは、問い合わせをした人のメールアドレスも入力してもらって、そこへメールで回答を送る形ですが、メールアドレスが間違っていると回答が届きません。そうならないように、メールアドレスを 2 回入れてもらうなどの対策をしますが、問い合わせる人にとっては面倒な話です。その点 maito を使ったリンクであれば、問い合わせがメールで来るので、回答を返信の形で必ず返すことができます。

4.2 PHP でメールを送る方法

PHP を利用してメールを送ることができます。そのために次のように `mb_send_mail()` を利用します。

```
mb_send_mail($to, $subject, $message, $headers, "-f $from");
```

$to にはメールを送る先のメールアドレスを入れておきます。$subject には件名を入れておきます。漢字を含む件名も可能です。$message には本文を入れておきます。本文に漢字を含むことは可能ですが、改行の位

置には「\n」を入れる必要があります。$from には差出人のメールアドレスを入れておきます。PHP が代わりにメールを送るために、本当のメールの差出人を設定しています。最後の$headers には次のようにして内容を入れます。

```
$headers="From: $from\n"
         ."Reply-To: $from\n"
         ."MIME-Version: 1.0 \n"
         ."Content-Type: text/plain; charset=ISO-2022-JP\n";
```

「From:」ではメールの差出人のメールアドレスを示します。「Reply-To:」ではメールの返信先を示します。大抵の場合は「From:」と同じメールアドレスになるので、この例では同じ変数を使っています。様々なところからメールを送るが、返信はどこか一つでまとまって受け取りたいと言う場合は、「From:」と「Reply-To:」に違うものを設定することになります。「MIME-Version:」と「Content-Type:」はメールの本文が日本語の場合、文字化けしないように入れておきます。

なお、mars の利用者のメールアドレスは次のような形です。

```
ユーザ名@mars.mgt.sugiyama-u.ac.jp
```

4.3 at を利用した予約実行

後の章に出てくる「IoT」や「スクレイピング」では、相手から定期的に情報を送ってくることもありますが、こちらから情報を取りに行かないとだめな場合も少なくありません。定期的に最新情報を収集し、そのまとめを利用者に Web ページで見てもらったり、こちらからメールでお知らせするという流れになります。ここでは、まず at コマンドを利用した単発の予約実行の方法を説明し、次に cron コマンドを利用した定期的な実行方法を説明します。

PHP で記述した内容をブラウザで呼び出すのではなく、その場で実行することができます。例えば date.php は次のような内容だったとします。

```
<?php
echo date("Y/m/d H:i:s"),"\n";
?>
```

mars に RDP で接続し、スタートメニューにある「システムツール」の中の「QTerminal」を選択すると、コマンド入力の画面が開かれます。そこで「wget -q -O - 」の後に date.php の URL を入力して Enter を押すと、date.php が実行されて現在の時刻が表示されます。なおコマンド入力の画面での貼り付けは、Ctrl+Shift+V となるのでご注意ください。

```
miki@mars:~$ wget -q -O - http://mars.mgt.sugiyama-u.ac.jp/miki/date.php
2021/09/21 16:30:16
miki@mars:~$
```

wget は指定した URL にアクセスして返ってきた内容を保存する Unix のコマンドです。「-q」オプションで余分なメッセージが出ないようにし、「-O -」オプションで返ってきた内容を表示するようにしています。これを指定した時刻に一度だけ行いたいときには次のようにします。ただし「miki」のところは自分のユーザ名に変えてください。

```
miki@mars:~$ at -t 10201345
warning: commands will be executed using /bin/sh
at> wget -q -O - http://mars.mgt.sugiyama-u.ac.jp/miki/date.php
at> <EOT>
job 5 at Wed Oct 20 13:45:00 2021
miki@mars:~$
```

1 行目の「at」が実行予約をする Unix のコマンドです。「10201345」は実行する時刻で、「10 月 20 日 13 時 45 分」を意味しています。月日時分は全て 2 桁で記述します。「at>」が表示されたらその後に実行するコマンドを入力します。3 行目に先程の実行する内容を指定しています。さらに続けて実行するものを入力することもできます。4 行目のところで Ctrl+D を押すと「<EOT>」と表示されて、実行される日時を表示して at コマンドの入力が終わります。

さてこのような方法で設定しても、その時刻に端末を利用しているとは限りません。そうすると実行した際に表示された時刻はどこに行くのでしょうか。at コマンドで予約実行した結果は、設定をしたユーザにメールで送られてきます。実行した際に何も出力がなければ、メールは来ません。mars ではスタートメニューの「インターネット」のところにある「Sylpheed」で自分に届いたメールを読むことができます。予定時刻が過ぎたら「Sylpheed」で結果を確認してください。

実は PHP のプログラムを端末で実行するだけならば、wget を使用して web サーバー経由で行う必要はありません。「wget -q -O -」の代わりに「php」だけで可能です。ただこのやり方は、at コマンドでは実行する主体が違うので、うまく動かない場合があります。それを回避する方法ももちろんあるのですが、割と複雑なので詳細は省略します。peditor で実行させてちゃんと動くことが確認できれば、端末で wget での実行を予約すると言う流れになります。

4.4 cron を利用した定期的実行

サーバーでは様々な処理が定期的に行われています。毎日、毎週、毎月と言う単位での定期的実行は Unix では cron と言うシステムが起動を行います。実は at コマンドの実行する部分も cron です。設定の仕方は大昔から変わっていませんが、少しわかりにくいところもあるので、設定用のアプリがないか少し探しましたが、なぜか無いようです。よって端末を起動して、設定ファイルを書き換えると言う形で設定します。

1. 「QTerminal」を起動します。
2. 「crontab -e Enter」と入力します。
3. 「nano」と言う編集プログラムが起動され、現在の cron の設定ファイルの内容が表示されます。「#」の記号からその行の終わりまではコメントなので cron の実行には関係ありません。
4. 定期的実行のための設定内容を入力・修正・削除します。
5. Ctrl+O で書き込み、Ctrl+X で終了します。
6. 定期的な実行の際に何か出力があれば、その内容がメールで届きます。

cron の設定は 1 件に付き 1 行になります。複数の設定は複数行になります。1 行の形式は次のようになります。

分　時　日　月　曜日　実行するコマンド

分〜曜日については、表 4.1 のようになっています。例えば毎週月曜日の 0 時 3 分にコマンドを実行するならば「3 0 * * 1 コマンド」、毎月 1 日の 0 時 15 分にコマンドを実行するならば「15 0 1 * * コマンド」、年中 10 分毎にコマンドを実行するならば「0,10,20,30,40,50 * * * * コマンド」となります。

表 4.1 cron の設定内容

	設定内容
分	*にすると毎分実行します。0〜59 の数値を一つか、複数を「,」で区切って並べます。
時	*にすると毎時実行します。0〜23 の数値を一つか、複数を「,」で区切って並べます。
日	*にすると毎日実行します。1〜31 の数値を一つか、複数を「,」で区切って並べます。
月	*にすると毎月実行します。1〜12 の数値を一つか、複数を「,」で区切って並べます。
曜日	*にすると全曜日に実行します。0〜6 の数値を一つか、複数を「,」で区切って並べます。0 が日曜日になります。

5.　IoT とグラフの表示

IoT (Internet of Things) は様々なものをインターネットに接続して利用しようと言う考え方で、それ自体新しい発想ではありません。ただ以前はネットワークへの接続にハードウェアやコストの面での問題が多く、それほど普及しませんでした。例えば 1990 年代後半だったと思いますが、この大学の演習室のパソコンを有線 LAN に接続するためには、各パソコンに 6 万円ぐらいする LAN ボードを別途購入する必要がありました。さらに LAN ボードを動かすためのドライバーソフトも各々インストールする必要がありました。現在ではノートパソコンならば無線 LAN、デスクトップのパソコンならば有線 LAN に接続するために何か追加で購入する必要はなく、ちょっとした設定のみでネットワークにつながります。この章で紹介する教室の温度・湿度・二酸化炭素を測定する機器 (以下環境センサー) で使われているコントローラーは、無線 LAN に接続するための機能を持ちながら 730 円 (税込み) です。

IoT については非常に多くの観点について学ぶ必要があります。既にある IoT の機器を利用するのであれば、多少楽になりますが、このような事が可能なのではないか？と思いつく土台になる知識となるとやはり大変です。ハードウェアが絡む話は女子大生向きではないかもしれませんが、ハードウェアが絡むと即座にはまねされないものにつながります。

この章では IoT に関するいくつかの観点の説明、教室の温度・湿度・二酸化炭素を測定する環境センサーを使って情報収集と処理、結果の表示方法としてグラフをどうやって出すのかについて述べたいと思います。

5.1　どうやってつなぐか

IoT ですから、当然インターネットにつながらないと困ります。ではなぜインターネットにつながる必要があるのか？それば物理的に遠いところにあっても、インターネット経由でほぼコストゼロでつながるからです。実際のところ IoT とは言うものの、近いところのものとつなげる話も少なくありません。敷地内にある工場にある機器、温室の温度計などで、歩いて行って見てくれれば済むではないかと言う見方もありますが、数が多い場合や記録を残す場合などに IoT にする利点が生じます。

インターネットにつながるコンセントがすぐ側にあるとか、無線 LAN の基地局の電波の届く範囲内にあるのであれば問題ありません。しかしそうでない場合もあります。公園のゴミ箱の空きを知ることができれば、ゴミ回収の合理化ができますが、公園の真ん中に LAN のケーブルは多分来ていません。また無線 LAN の基地局も期待できません。幸いなことに携帯やスマホのおかげで、これらが使用する無線回線は山奥などでなければ利用することができます。ただお金がかかります。河川が溢れていないか動画で 24 時間監視すると言うことになると、スマホの回線の通信料金は一番高いプランになるでしょう。一方公園のゴミ箱の空きを知るためにゴミ箱の内部の動画は必要ないでしょうし、ゴミ箱が溢れてもすぐに回収には行けませんので、1 時間に 1 回程度ゴミの量が分かれば良いでしょう。幸いなことにそのような通信量が少ない用途に対する通信サービスがあります。例えば LPWA (Low Power Wide Area-network) を利用した Sigfox[*3]と言うサービスでは、年額 1,500 円程度で 1 回に付き 12 バイトのデータを 1 日最大 140 回送ることができます。これらを比較すると表 5.1 のようになります。

表 5.1　接続方法の比較

接続に利用するもの	データ量	料金	接続設定
現場にある LAN や無線 LAN	動画も可能	無料 (既存設備があれば)	現場に合わせる
携帯やスマホの回線	動画も可能	高価格 (データ量が多いと特に)	事前設定可能
Sigfox など	極めて少ない	低価格 (データ量が少ないと特に)	事前設定可能

[*3] Sigfox は世界的に展開されているサービスで、国内では京セラコミュニケーションシステムが扱っています。https://www.kccs-iot.jp/service/

データ量の多少が大きな分かれ目になります。現場にあるネットワーク環境を利用する方法は安上がりですが、そのネットワーク環境に依存する接続設定になるので、セットアップに分かる人が必要になります。他の 2 つの方法はお金がかかりますが、製造時に接続設定ができるので、特に個人向けの機器ではすぐ使えるので便利となります。

5.2 電源をどうするか

IoT の機器も電気がなければ動きません。電灯線から交流 100V の供給を受ける、電池で動かす、太陽電池パネルなどで自家発電するなどが考えられます。電灯線は、停電と電気工事が必要になると高く付くという問題があります。一方電池のように、空にならないように管理する必要がありません。電池ではスマホのように毎日充電が必要では手間がかかりすぎるので、できるだけ機器の消費電力を減らして電池が長持ちするようにします。1 年半ぐらい電池が持つのであれば、年に 1 回電池を交換するという運用で使えます。それでも Sigfox の通信料より電源の方が費用がかかる恐れがあります。太陽電池で発電し、電池を充電して使うような形であれば、電池の心配をしなくて済みます。ただお天気が悪い日が続いたり、室内など陽が当たらない場所では困ります。太陽系の果に行くような人工衛星では、太陽電池が使えないので原子力電池を使うそうです。

5.3 センサーをどうするか

インターネット経由で伝える情報をどのようにして入手するかも大きな問題です。情報は電気信号の形でないと送れません。物理的な信号を電気信号に変換するものをセンサーと呼び、現在多くの種類のセンサーがあります。例えば「Interface」と言う雑誌の 2021 年 7 月号別冊付録の「IoT センサ図鑑」には、温度、湿度、気圧、動き、力、距離、接近、匂い、ガス、色、位置、電流、生体検知、音、タッチのセンサーが紹介されています。温度センサーと言っても、測定できる温度の範囲、精度、出力形式が異なるものが色々あります。高精度なものほど高価になります。気圧センサーで高精度なものでは、5cm 程度の高さの違いによる気圧の変化が分かるものもありますが、そこまでの精度が必要な場合は限られるでしょう。重さのセンサーがありませんが、力のセンサーを用います。重さがかかると形が変わるものに力のセンサーを貼り付けて、形の変わり具合から重さを測ります。このように違うセンサーを用いて測定することがあります。例えば電気機器は電気で動きますので、電流センサーで機器に流れる電流を測れば、機器が動いているかどうか分かります。

AI で処理をすることにより、高度な判定をするセンサーになることがあります。防犯対策用にコンビニの棚を撮影するカメラ画像から、棚の空き具合を AI で推定するという話があります。棚が空いたままでは売れませんので、空いている棚をセンサーで検出し、すぐに商品を補充することによって売上が増えたそうです。このような AI の処理を IoT の方でやってしまうか、データを集めてからデータ処理でやるかが問題になります。画像データと棚の空き具合の数字となると、データ量がかなり違うのでカメラのところで推定するのが望ましいです。その代わりカメラのところに AI を実行するハードウェアが必要になります。棚の画像であれば、棚や商品が文句を言うことがないので良いのですが、人の姿を撮影して処理をするような場合は問題になる可能性があります。そのような場合カメラのところで処理をして、サーバーには個人情報にはなりにくい認識結果だけ伝わるようにする、と言うような配慮が求められることがあります。

5.4 集めた情報の処理

センサーから集めた情報は大量になる事も多く、ビッグデータと呼ばれることもあります。大量のデータを人が個々に見ると言うわけにも行かないので、統計的な処理を行うのが普通ですが、なかなかうまく行きません。処理を行って何を得たいのかを明確にし、それに合った統計的手法を選択するのが一つの方法です。漠然と様々な統計的手法を使ってみるのでは、時間や費用ばかりかかります。

IoT でゴミ箱のゴミ量を測定して、一杯になったゴミ箱だけゴミを回収をすることにより、回収費用の節減

やゴミが溢れてゴミ箱が使えなくなることを防ぐ、とします。送られてきたゴミ量から今ゴミ箱が溢れているかどうかの判定は容易でしょう。でも溢れたからと行って即座にゴミ回収には出発できないでしょうし、回収の順番によっては回収できるのはだいぶ先になることもあるでしょう。そうすると送られてきたゴミ量から、回収に行ける頃にちょうど溢れるゴミ箱が分かればよいことになります。そうなると過去のデータから予測ができるようにしなければいけません。ゴミ箱の設置場所、時間帯、曜日、季節などによって捨てられるゴミの量は変わるでしょう。予測ができるようになるまでに、それなりの量のデータが必要になります。また曜日や季節は、センサーによって得られる情報ではありませんが、より正確な予測には欠かせないと思われます。

これからゴミを回収すべきゴミ箱がきちんと予測できたら終わりではありません。できるだけ短い経路で回らないと時間や燃料の無駄が生じます。ところがこの最適な経路を求める問題は、古くから「巡回セールスマン問題」と呼ばれるもので、ゴミ箱の数が数十個のレベルでも莫大な計算量[*4]が必要となり現実的な時間で正解[*5]は得られません。現実的には最適解の 1.5 倍以上は悪くならないけど高速に求まる、と言うような解法があるのでそちらで我慢するとか、量子コンピュータであればすぐ答えが出るとか言われています。

5.5　環境センサーからの情報の取り込み

IoT 機器の多くは節電のために、適当な時間間隔でサーバーにデータを送ってきます。サーバーから要求を送ってデータをもらう形にすると、IoT 機器は常に要求待ちのために電力を消費します。送信の時だけ目覚めるようにして、ほとんどの時間を死んだようになって節電するようにして、時には数百倍も電池の寿命を延ばします。

今回利用する環境センサーは 5 秒毎に気温、湿度、二酸化炭素濃度、明るさを測定して表示します。そして 20 秒毎に mars にデータを送ります。mars は送られてきたデータに時刻情報を追加したものをデータベースに保存します。実際の IoT によるセンサー情報の収集についても同じように、センサー情報をまとめるサーバーがよく使われます。まとめるサーバーは処理能力は不要ですが、ダウンするとその間のセンサー情報が欠落するので、サーバーが停止しないように十分な対策を行います。そして別のサーバーで集めたデータの処理、その結果をまた別の Web サーバーから公開するという形になります。

手順としては次のようになります。

1. データを受け取るプログラムを作成します。データは POST 方式で送られてくるので、`$_POST['date']` で時刻、`$_POST['room']` で環境センサーのある部屋の名前、`$_POST['temp']` で室温、`$_POST['humi']` で湿度、`$_POST['CO2']` で二酸化炭素の濃度、`$_POST['bright']` で明るさを受け取ることができます。受け取ったデータはファイルへ入れるか、データベースに入れましょう。
2. プログラムを登録する際は `http://mars.mgt.sugiyama-u.ac.jp/Sensor/` へアクセスします。「送り先の設定」のリンク先の「送り先登録」のところに、作成した受信プログラムの URL の一部を入力して、登録をクリックします。登録を削除したい場合は、同じく受信プログラムの URL の一部を入力して、削除で削除できます。
3. 最初のページに最近受信したデータが表示されるので、これと比較することによって自分のプログラムが正しく受信できたかどうか確認することができます。表示内容は自動的に更新はされないので、適宜ブラウザの更新ボタンをクリックしてください。

環境センサーの時刻は特に校正されていないので、20 秒毎と言っても 1 分間に必ず 3 回データが来るとは限りません。よって 2〜4 回データが来るものとして、時刻情報をもとに 1 分間の平均を記録するようにします。割り算は難しいと言うのであれば、一番最初か最後のデータを採用すると言う手もあります。一般的には平均が望ましいとされますが、平均は極端な値に引きずられるという性質があります。つまりノイズによる小

[*4] ゴミ箱の数が n ならば n!に比例する時間がかかります。
[*5] 例えば明日の天気予報の計算に、1 日以上かかるのでは意味がないですよね。

さな変動は平均して均すことによって減らすことができますが、センサーの誤動作による極端な値が混ざった場合はうまく行きません。よく用いられるのは 5 つの測定値から、最大と最小を除いた 3 つの平均を採用すると言う方法です。また中央値を採用することもあります。

5.6 グラフの描画

web ページにグラフを描く方法は色々あります。いつも同じ内容のグラフで良ければ、Excel でグラフを作成し、それを画像ファイルにして、HTML の Img タグで表示しても良いわけです。ただこの方法は、内容を変更するには人手が必要です。IoT などを利用して刻々と変わるデータをグラフの形で示したい場合は、次のような方法が考えられます。

- PHP でグラフ画像ファイルを作成する。その画像を HTML の Img タグで表示する。
- PHP でグラフ作成に必要なデータを用意し、JavaScript でそのデータをグラフの形で表示する。

データの更新が 1 日に 1 度程度であれば、PHP でグラフ画像を作成するのが良いでしょう。グラフ画像ファイルを残しておけば、アクセスが多い場合も再処理の必要がありません。また JavaScript の実行に問題があっても、画像の表示に問題があるブラウザはないと思います。毎分のようにデータが更新されるような場合は、グラフ画像を残しても無駄になる可能性が高いので JavaScript でも良いかもしれません。グラフの画像ファイルと JavaScript のファイルのどちらが大きくなるかは場合によりますが、大きなグラフでは画像ファイルの方が大きくなるかもしれません。

ここでは JavaScript を使用する方法を紹介します。既に多くの JavaScript によるグラフ作成のためのライブラリが公開されていますが、ここでは比較的紹介する web ページの多い「Chart.js」[*6]を取り上げます。

1. `<Head>`と`</Head>`の間に以下を入力します。

```
<Script src="chart.min.js"></Script>
```

   これで chart.js のコメントなどを除いたものを取り込みます。`https://cdnjs.com/libraries/Chart.js` によると、現在 (2022 年 11 月) の最新のものは 4.0.1 でした。より数字の大きいものの方が、新しいのでバグがなくなり、機能が増えていると思いますが、使い方が変わる事もあるので、以下の記述は ver. 3 のものです。最新のものを無理に追い求める必要はないでしょう。また検索すると使用例が数多く見つかりますが、バージョンが違う例には注意しましょう。

2. グラフを表示するところに以下を入力します。

```
<Div style="width: 400px; height: 200px">
<Canvas id="myGraph"></Canvas>
</Div>
```

   グラフの大きさを Div の方で設定します。Div がない場合は、ブラウザの横幅一杯のグラフが表示されます。グラフの縦横比は折れ線グラフや棒グラフでは 1:2 になります。四角いグラフを作成したい場合は Canvas の方に「 width=”200” height=”200”」のような指定を追加します。「myGraph」の部分は適当な名前で構いません。ただし次に出てくる document.getElementById のところで同じ名前を指定する必要があります。また一つの web ページに複数のグラフを入れる場合は、必ず相異なる名前にします。

[*6] `https://www.chartjs.org/`、「とほほの WWW 入門」の「ライブラリ」のところに説明のページがあります。`https://www.tohoho-web.com/ex/chartjs.html`

3. Canvas タグの後に以下のようなグラフの定義を入れます。

```
<Script type="text/javascript">
   new Chart(document.getElementById("myGraph"), {
      type: タイプ,
      data: { データ },
      options: { オプション }
    });
</Script>
```

「タイプ」のところに表 5.2 のようなグラフの種類が入ります。この他にはレーダーチャート、散布図、ドーナツチャート、鶏頭図[*7]、バブルチャート、面グラフ、混合チャートなどが可能なようです。

表 5.2　グラフの種類

type に指定するもの	意味
'line'	折れ線グラフ
'bar'	棒グラフ
'circle'	円グラフ

「データ」のところには次のような形でグラフのデータが入ります。

```
labels: [ 見出しのデータ ],
datasets: [
     { 1 つ目の系列のデータ },
     { 2 つ目の系列のデータ }
],
```

「見出しのデータ」の部分には x 軸の下に並ぶものが入ります。例えば「'1 月', '2 月', '3 月', '4 月'」のようなものです。「1 つ目の系列のデータ」のところには、次のような形で系列のデータが入ります。もし線が一本の折れ線グラフであれば「2 つ目の系列のデータ」の部分は不要ですし、線が 3 本以上の折れ線グラフでは同じ形で「3 つ目の系列のデータ」を追加します。

```
label: [ 系列の見出し ],
data: [ 系列の数値 ],
色の指定
```

「系列の見出し」には、例えば「' 売上'」のようなものが入ります。また「系列の数値」には、例えば「100, 150, 90, 120」のように「,」で区切った数値が並びます。「色の指定」は表 5.3 のような指定を必要な数だけ「,」で区切って並べます。
「色」の部分には「'rgba(255,0,0,1)'」のように赤 (0〜255)、緑 (0〜255)、青 (0〜255)、透明度 (0〜1) を数字で指定する方法があります。透明度は 0 の時は完全に透明なので背景の色のままになります。0.5 にすると指定した色と背景の色が半々に混ざります。1 にすると背景に関係なく指定した色になります。指定した色にするのであれば、「'rgb(46,106,177)'」のようにしてもよいし、HTML と同様に 16 進数で「"#BB5179"」のように指定することもできます。
「オプション」のところにグラフの形状などの細かい設定が入ります。

[*7] 円グラフの一種だが、量の大きさを扇形の開く角度ではなく、半径の長さで示すもの。

表 5.3 グラフの色の指定の例

指定例	意味
borderColor: 色	線の色の指定。棒グラフでは塗りつぶす色より濃い色を指定すると綺麗。
backgroundColor: 色	塗りつぶす色の指定。半透明にすると綺麗。円グラフでは「,」で区切って色を数字の数だけ指定する
borderWidth: 1	線の太さを数字で指定する

```
plugins: {
   title: {
      display: true,
      text: グラフのタイトル
   }
},
scales: { 軸の設定 }
```

「グラフのタイトル」のところは、「' 売上の推移'」のような形で指定します。タイトルが不要な場合は、「display: false」にします。折れ線グラフや棒グラフでは「軸の設定」をしますが、円グラフには軸がないので、「scales: 〜」を丸ごと省略して構いません。「軸の設定」は次のような形になります。

```
y: {
   max: 目盛りの最大値,
   min: 目盛りの最小値,
   ticks: {
      stepSize: 目盛りの刻む幅,
      callback: 目盛りの数値の出し方
   }
}
```

「y」が縦軸を示しています。散布図では横軸も「x」として指定します。「目盛りの最大値」は縦軸の一番上の数字、「目盛りの最小値」は縦軸の一番下の数字、「目盛りの刻む幅」は例えば 10 を指定すれば、10、20、30 と言うような目盛りになります。これらの設定は Chart.js にお任せでよければ全て省略可能です。「目盛りの数値の出し方」は、数字に何かを付けて目盛りに出したい場合に指定します。例えば数字に「円」を付けて出したいのであれば、「`function(value,index,values){return value+'円';}`」のように指定します。

説明が重層的で分かりにくいので簡単な例を示します。リスト 1 は折れ線グラフの例です。これによって図 5.1 のようなグラフになります。

リスト 1 折れ線グラフの例

```
1 <HTML lang="ja">
2 <Head>
3 <Meta charset="UTF-8">
4 <Title>折れ線グラフのテスト</Title>
5 <Script src="chart.min.js"></Script>
6 </Head>
7 <Body>
8 <Div style="width: 600px; height: 300px; border: 1px solid black;">
9 <Canvas id="myGraph"></Canvas>
```

```
10 </Div>
11 <Script type="text/javascript">
12    new Chart(document.getElementById("myGraph"), {
13       type: 'line',
14       data: {
15          labels: ['9/1', '9/2', '9/3', '9/4', '9/5', '9/6', '9/7'],
16          datasets: [
17             {
18                label: '最高気温',
19                data: [31.7, 27.7, 24.7, 26.4, 30.7, 31.0, 27.3],
20                borderColor: 'rgba(255,0,0,1)',
21                backgroundColor: 'rgba(255,0,0,0.5)'
22             },
23             {
24                label: '最低気温',
25                data: [24.8, 24.8, 22.9, 22.2, 21.8, 21.9, 21.6],
26                borderColor: 'rgba(0,0,255,1)',
27                backgroundColor: 'rgba(0,0,255,0.5)'
28             }
29          ]
30       },
31       options: {
32          plugins: {
33             title: {
34                display: true,
35                text: '2021年　名古屋市の気温',
36             }
37          },
38          scales: {
39             y: {
40                max: 35,
41                min: 15,
42                ticks: {
43                   stepSize: 5,
44                   callback: function(value, index, values){ return value+'度'; }
45                }
46             }
47          }
48       }
49    });
50 </Script>
51 </Body>
52 </HTML>
```

リスト 1 について補足すると：

- 8 行目の Div に border の設定を付けて、グラフの周りに線が引かれるるようにしています。
- 15 行目のところは固定であればこのままで良いのですが、データによって変わるのであれば PHP で出すようにします。もし$date に「'9/1', '9/2', '9/3', '9/4', '9/5', '9/6', '9/7'」のようなものが入っていれば、次のような感じです。

```
    labels: [<?= $date ?>],
```

- 19 行目と 25 行目の数値の部分も実際の利用の際には、PHP でその内容を出すことになるでしょう。
- 20 行目で色として赤の成分しか指定していないので、線は赤になります。透明度も 1 なので背景の色の影響はありません。
- 21 行目では透明度が 0.5 なので、背景の影響を受けて薄い赤になります。折れ線グラフではこの色は凡例のところにしか使われていませんが、棒グラフではこの色が棒の色になります。

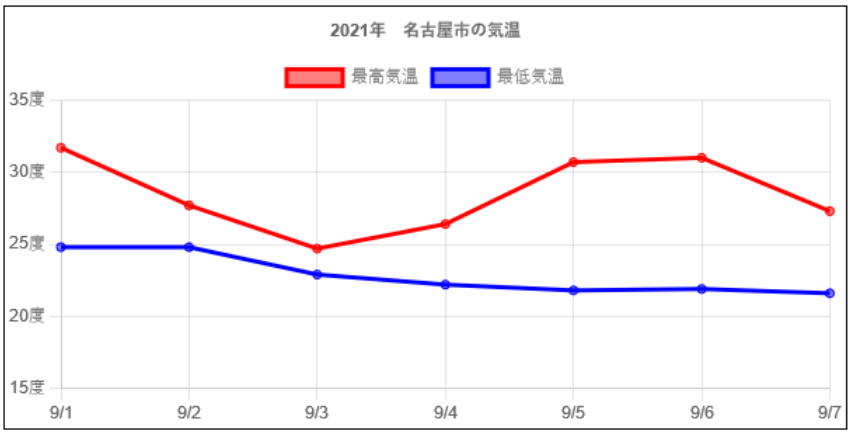


図 5.1 折れ線グラフの例

5.7 グラフ化するデータの作成

IoT を利用してデータを自動的に収集するとすぐ膨大なデータになります。環境センサのデータは 1 分間に 1 件しか来ませんが、1 時間で 60 件、1 日で 1,440 件、1 週間で 10,080 件、1 ヶ月で約 43,200 件、1 年で約 525,600 件になります。グラフでこれを示す場合、目的に合わせてデータの一部を用いるのが普通です。例えば大学では毎週同じ曜日に同じ科目が同じ教室で行われます。よって 1 週間分のグラフを見ることにより、人数が多くて二酸化炭素濃度が高くなる授業が分かるでしょう。そこでその授業の教員に換気に注意するように伝えて、その効果を確認することを考えると、前の週のグラフも見えるようにすると良いでしょう。

またグラフを表示する際のドットの数よりデータの数が多い場合は、データをまとめることを考える必要があります。1 週間のグラフの場合約 10 万件ありますので、幅が 500 ドットならば 200 件を 1 つのデータ (代表値) にする必要があります。代表値としては平均がよく使われますが、最大値、最小値、中央値、最頻値などの方が良い場合もあります。

部分的に取り出す SQL 文

主キー (id) が連続している場合を仮定すると、id の値の範囲を条件にすることによりデータの一部を取り出すことができます。例えば次のようにします。

```
SELECT * FROM sensor WHERE id BETWEEN 100 AND 199
```

これで id が 100〜199 の 100 件のデータが取り出せますが、 それが想定した時間の幅に対応するかどうかわ分かりません。機器やネットワークのトラブルなどでデータに欠損がある場合、もっと長い時間に対応する 100 件のデータになります。例えば毎分 1 件収集したデータであれば、100 件のデータは通常 100 分に対応しますが、途中トラブルのため 3 日間データが収集できなかった場合は 3 日 +100 分に対応します。

よって毎分 1 件のデータを 1 日分取り出したい場合、id で 1,440 件と言う範囲を指定するより日付のデータがあればそれを利用した方が良いでしょう。date に通算秒数が入っており、1669474800 は 2022 年 11 月 27 日 0 時 0 分 0 秒の通算秒数、24 時間は 86,400 秒なので次のようにして 2022 年 11 月 27 日の 1 日分のデータを取り出すことができます。

```
SELECT * FROM sensor WHERE date BETWEEN 1669474800 AND 1669474800+86400-1
```

ついでに co2 に二酸化炭素の濃度が入っていおり、2022 年 11 月 27 日の 1 日分の二酸化炭素濃度の平均値を求めたい場合は、次のようにして求めることができます。

```
SELECT AVG(co2) AS CO2 FROM sensor
                        WHERE date BETWEEN 1669474800 AND 1669474800+86400-1
```

平均値は「CO2」と言う別名を付けているので、この名前で取り出します。残念ながら`$data['AVG(co2)']`のようなことはできません。1 日分の平均を 100 個使ってグラフを作るという場合は、この検索を 100 回繰り返すことになりますが、何度もグラフを作るのであれば、1 日分の平均を別のテーブルに入れておいた方が良いでしょう。その場合グラフを作る際でなく、受け取ったデータが次の日のデータに変わった時点で前の日の平均を別のテーブルに入れる方が良いでしょう。

余り多くの件数のデータの平均を求めるのでなければ、次のように毎回 SQL でやっても良いでしょう。これは 1 日分のデータから 30 分毎の平均を求めています。

```
SELECT date, AVG(co2) AS CO2 FROM sensor
                      WHERE date BETWEEN 1669474800 AND 1669474800+86400-1
                      GROUP BY date/1800
```

「date/1800」で通算秒数の date を 30×60 で割ることにより、date を 30 分単位の数値に直しています。date は INTEGER (整数値) なので割り算の結果も小数点以下が切り捨てられます。その結果同じ数値になるデータが同じグループになり、グループごとの二酸化炭素濃度の平均が求められます。1 日の平均二酸化炭素濃度と違って、こちらは 48 個の平均値が出てくるので、date の値も取り出しています。同じグループに属する 30 件のデータのどの date が出て来るのかですが、グループ内の一番最後のデータの date が出てくるようです。

SQL で検索条件や並び替えを行なって得られた結果の件数が多すぎて一つの表やグラフに入り切らない場合、他のページに続きを出すことになります。例えば最初のページに 100 件出したので、続きのページは 101 件目から出したい場合です。この時検索や並び替えがあるので id は当てになりません。次の例は二酸化炭素濃度の大きい順に最初の 100 件を取り出すものです。

```
SELECT * FROM sensor ORDER BY co2 DESC LIMIT 100
```

LIMIT の後の数字で取り出す件数を指定します。もしこれより少ない件数しかなかった場合は、その全てが出てくるだけでエラーにはなりません。さらに 101 件目以降を 100 件取り出したい時は次のようにします。

```
SELECT * FROM sensor ORDER BY co2 DESC LIMIT 100 OFFSET 100
```

OFFSET の後の数字で飛ばす件数を指定します。もしこれより少ない件数しかなかった場合は、何も出ないだけでエラーにはなりません。

5.8 超えたらメールを送る

センサーから送られてきた値に何か問題があった場合は、それを誰かに知らせなければなりません。既に取り上げたメールを使って知らせるのが一つの方法です。問題があるかどうかの判定は、複数の条件が複雑に絡まっている場合もありますが、単純に数値が基準を上回ったら問題という例が多く見られます。例えば河川の水位が危険水位を超えたら避難しなければなりません。環境センサーの場合も、二酸化炭素濃度が例えば 2,000ppm を超えたら「急いで換気しましょう」と言うメールを送るなどが考えられます。もし教室にある換気扇がネットワークにつながっていてリモートコントロール可能であれば、メールを送る代わりに換気扇を動かした方が良いのでしょうが。

「2,000ppm を超えたら」と言う条件の if 文でメールを送るという方法では問題を生じることがあります。まず毎分新しいセンサーの結果が来るので、10 分間 2,000ppm を超えたら 10 通メールを送ることになります。状況が改善されるまで、ひたすらメールを送るのは多分迷惑です。そこで一度 2,000ppm を超えたらメールを送り、その後超えたままの場合はメールを送らないようにする必要があります。ところがセンサーの数字は通常微小なふらつきを伴います。2,001 → 1,999 → 2,001 → 1,999 → 2,001 のように境界をまたぐ形でフラフラするとその度にメールが行くことになります。

このような問題に対処するために基準を 2 つ用意します。例えば 2,000ppm を超えたら問題状態、1,900ppm を下回ったら問題状態解除のようにします。これならば一度 2,000ppm を超えて問題状態となれば、センサーの数字が多少ふらついても問題状態解除にはなりません。ただセンサーから 1,950ppm と言う数字が届いた場合、今は問題状態なのかはこの数字だけでは判断できません。つまりこれまで問題状態であれば問題状態ですし、これまで問題状態でなければ問題状態ではありません。ずっと 1,900ppm〜2,000ppm の間をふらふらしていた場合は、それ以前まで遡って調べないと判定できません。毎回遡って調べるのも面倒なので、センサーの値を入れるテーブルに問題状態かどうかも入れておくと良いでしょう。

5.9 web ページの自動更新

通常の web ページは一旦表示されると利用者が何かしない限り動くことができません。よって IoT から随時データが更新されても、それを伝えることができません。ここでは比較的容易に実現できる web ページの自動更新の方法を紹介します。これ以外には、web サーバーとつながったままにするとか、JavaScript でこっそり web サーバーからデータをもらう方法などがあります。

ブラウザから 10 秒毎に更新の要求を出させるには、`<Head>`と`</Head>`の間に次のような指示を入れます。

```
<META HTTP-EQUIV="Refresh" CONTENT="10">
```

これの応用として、次のようにすると 10 秒後に別のページを見せることも可能です。

```
<META HTTP-EQUIV="Refresh" CONTENT="10;URL=別のページの URL">
```

これらを PHP で行う時は、次のように header() を使用します。通常のタグ等の出力より先行して実行する必要があるので注意します。こちらはブラウザでソースの表示を行っても出てきません。

```
header("Refresh: 10");
```

```
header("Refresh: 10;URL=別のページの URL");
```

6. スクレイピング

スクレイピング (Web scraping) は既存の Web ページから情報を取り込む技術のことです。情報を取り込むには、

1. web ページの内容を取り込む
2. 取り込んだ内容から必要な部分を取り出す

ができる必要があります。その際に問題となるのは、

- URL が「https://」で始まる場合、SSL の設定ができていないと内容を取り込めない。mars の場合は既に SSL が設定されているので問題ありませんが、ちょっと面倒な話です。
- URL が決まっていない場合がある。毎日新しい情報を提供している場合、日付が URL の中に埋め込まれていることがあります。このような場合、大抵入り口となる web ページは固定の URL となっているので、そこにまずアクセスして、今日の URL を求める必要があります。
- 予告なしに URL が変わることがあります。そして従来の URL へのアクセスができなくなるとエラーが出るので気がつくのですが、従来の URL へのアクセスは可能だが内容の更新がされなくなる、と言うような場合は気づきにくいので困ります。
- ページの構造が時々変わる場合がある。コロナのワクチン接種状況のページが、医療関係者だけ、高齢者も追加、一般の人も追加と対象が広がるに従って変わった例があります。さらにその際に URL も変更になってました。
- 必要な情報が HTML のタグの中に埋もれているので、探し出さなければならない。唯一のタグに囲まれていればすぐに見つかりますが、大抵は無数にある`<Td>`〜`</Td>`や`<Div>`〜`</Div>`の一つに囲まれているのが普通です。
- 必要な情報が JavaScript の実行により動的に生成されている場合は、JavaScript のソースが得られても情報自体は得られません。このような場合は、ブラウザを動かしてその表示内容を取り出すような大変高度な技術が必要になります。

このような様々な技術的な問題もありますし、著作権の問題や多くの人がスクレイピングを行ったために、スクレイピングが禁止となったサイトや情報提供をやめてしまったサイトもあります。通常のブラウザによるアクセスと変わらないような頻度や量であれば問題にはなりにくいと思います。

6.1 Web ページの取り込み

PHP で web ページの取り込みをするのは、file 関数を用いれば簡単にできます。

```
$lines=file("https://www.mgt.sugiyama-u.ac.jp/");
foreach ($lines as $line) {
   echo str_replace("<","&lt;",$line),"<Br>\n";
}
```

オプションとして「FILE_IGNORE_NEW_LINES」を file() で指定すると一つの変数に取り込まれますが、この例のように指定しない場合は、取り込んだ結果は配列型変数になるので、foreach を利用して 1 行ずつ取り出しています。取り出したものをそのまま echo で表示すると、含まれている HTML のタグが働いてしまうので、「`<`」を全て「`<`」に置き換えています。

ブラウザで URL を入力するだけで表示される web ページなのに、このやり方では「HTTP request failed!」で拒絶されるところが増えています。そのような場合は、リスト 2 のようにブラウザのふりをすると応答してくれることがあります。file() では情報が追加できないので、file_get_contents() を使用しています。

リスト 2 ブラウザのふりをする例

```
1  $url="https://sample.php";
2  $options = array(
3     'http' => array(
4        'header' => "User-Agent: Mozilla/5.0 (Windows NT 6.2) AppleWebKit/537.1 ".
5                    "(KHTML, like Gecko) Chrome/21.0.1180.75 Safari/537.1"
6     )
7  );
8  $page=file_get_contents($url,false,stream_context_create($options));
```

一方何か入力をして「送信」をクリックしたら、結果の web ページが出てくるような場合があります。入力内容を method として GET で送っている場合は、その内容が URL に反映されるので良いのですが、POST で送っている場合や cookie を利用している場合 (SESSION 変数を使用している場合も含む) は、これらの情報を付けた上で取り込まないと、エラーメッセージのみが返ってくる場でしょう。リスト 3 はそのような web ページへの対応方法の例です。

リスト 3 POST のデータや cookie を送る例

```
1  $url="https://sample.php";
2  $param=array(
3     'id' => $id,
4     'pass' => $pass);
5  $options = array(
6     'http' => array(
7        'method' => 'POST',
8        'header' => "Cookie: ".$cookie."\r\n".
9           "User-Agent: Mozilla/5.0 (Windows NT 6.2) AppleWebKit/537.1 ".
10          "(KHTML, like Gecko) Chrome/21.0.1180.75 Safari/537.1\r\n".
11          "Content-type: application/x-www-form-urlencoded",
12       'content' => http_build_query($param),
13    )
14 );
15 $page=file_get_contents($url,false,stream_context_create($options));
```

- 3〜4 行目で POST で送るデータの用意をしています。「id」と「pass」が名前で、内容は$id と$pass に入っていると仮定しています。
- ８行目で$cookie に入っているものを cookie として設定していますが、$cookie の内容は、その前のアクセスの際の web サーバーからの応答の中から取り出しておく必要があります。cookie を使用していない場合は、「`"Cookie: ".$cookie."\r\n".`」の部分は削除します。応答から cookie を取り出す方法の詳細は省略しますが、`file_get_contents()` を使用してアクセスした後に`$http_response_header` という変数に相手から送られてきたヘッダー情報が入るので、この変数から取り出します。
- 9〜10 行目ではブラウザを偽装しています。これがないとエラーを返してくるサイトがあります。
- 15 行目で$page に web ページの内容が入ります。

6.2 HTML の解析

変数の中に取り込んだ web ページの内容から必要な情報を取り出すためには、情報のある場所を示すものを目印にします。例えば「現在のポイント数」と言う文字の後に取得したい情報があるならば、mb_strpos() などで「現在のポイント数」の位置を求めます。また web ページの内容には多数の HTML のタグが含まれてい

ます。これを目印に取り出すことが考えられます。目的の情報が 3 つ目の表の 1 行目にあるならば、3 つ目の「<Table>」の後の最初の「<Tr>〜</Tr>」にあるはずです。これを単純に「<Tr>」だけを目印に探すのは無理で、「<Table>」を探し、次に「<Tr>」を探すというようなステップを踏む必要があります。

web ページの内容を HTML のタグに分解してくれるライブラリがあります。ここでは「PHP Simple HTML DOM Parser[*8]」を紹介します。他にも「phpQuery[*9]」などがあります。「PHP Simple HTML DOM Parser」を利用するには、まずこのライブラリをダウンロードする必要があります。「https://sourceforge.net/projects/simplehtmldom/files/simplehtmldom/」をブラウザで開くと、一覧が表示され様々なバージョンのものがあることが分かります。2023 年 12 月の時点では「2.0-RC2」が最新のようですが、アクセス数を見ると「1.9.1」の方が多いのでそちらにします。「1.9.1」をクリックするとこのバージョンのページが表示されるので「simplehtmldom_1_9_1.zip」をクリックしてダウンロードします。Windows の場合ダウンロードしたファイルを右クリックしてメニューで「全て展開」を選択すると解凍することができます。解凍してできたフォルダーの中にある「simple_html_dom.php」を peditor でアップロードします。

「PHP Simple HTML DOM Parser」の簡単な使用例はリスト 4 のようになります。

リスト 4　PHP Simple HTML DOM Parser の簡単な例

```
1  <HTML lang="ja">
2  <Head>
3  <Meta charset="UTF-8">
4  <Title>簡単な例</Title>
5  </Head>
6
7  <Body>
8  <?php
9  include "simple_html_dom.php";
10 $page=file_get_contents("https://www.mgt.sugiyama-u.ac.jp/");
11 $html=str_get_html($page);
12 echo $html->find("body",0)->plaintext;
13 ?>
14 </Body>
15 </HTML>
```

- 9 行目でライブラリのファイルを取り込みます。
- 10 行目は web ページの内容を$page に取り込んでいます。ここの部分は他のサイトのページを取り込む場合、前章の内容と同様に変更する必要があります。
- 11 行目で web ページの内容を HTML のタグで分解したものを$html に入れています。
- 12 行目では 1 番目の「Body」タグの内容を取り出し、その中の文字の部分のみを取り出して、echo で表示しています。find() の中のゼロが 1 番目を示しています。

「Body」タグの中の「H1」タグの中身を取り出したい場合は次のようにします。

```
echo $html->find("body",0)->find("h1",0)->plaintext;
```

このように find() を重ねることにより入れ子になった HTML のタグの中の方にあるものを取り出すことができます。また HTML のタグで挟まれたものではなく、タグの中で指定したものを取り出すこともできます。

[*8] https://simplehtmldom.sourceforge.io/
[*9] https://code.google.com/archive/p/phpquery/downloads

```
echo $html->find("body",0)->find("a",0)->href;
```

これで A タグで指定した href の値、つまりリンクの URL を取り出すことができます。途中の HTML のタグは、次のように省略することができます。ただしその場合何番目かの数字が変わる可能性があります。

```
echo $html->find("a",0)->href;
```

Div タグはよく使われていますが、その中身に合わせて id や class の指定がされていることがよくあります。例えば「`<Div id="menulist">`」の中の文字を表示するならば、

```
echo $html->find("div[id=menulist]",0)->plaintext;
```

のようにします。class の場合も同様です。

次のように foreach と組み合わせて、同じタグの内容を全て取り出すことも可能です。foreach の中の find() の中の指定がタグだけになっているところがこれまでの使い方と違います。タグだけ指定すると、指定されたタグを全て取り出します。そして foreach で全てから 1 つずつ取り出しているのです。

```
foreach($html->find("a") as $e) {
   echo $e->href,"<Br>\n";
}
```

これで web ページ中の A タグに指定された URL が全て表示されます。

同じタグの内容を全て取り出すのではなく、出てきたタグを順番に扱いたい場合はリスト 5 のようにします。

リスト 5 タグを出てきた順に扱う例

```
1 $parent=$html->find('h3',0)->parent();
2 foreach ($parent->children() as $tag) {
3    if ($tag->tag=='h3') {
4       H3 のタグに対する処理
5    }
6    if ($tag->tag=='table') {
7       Table のタグに対する処理
8    }
9 }
```

1. 1 行目では 1 つ目の H3 タグの親となるタグを$parent に入れています。`<X>〜<Y>〜</Y>〜</X>`のようにタグ X の中にタグ Y が含まれる時、タグ Y の親はタグ X になります。またタグ X の子はタグ Y になります。表示されているタグを全てという場合は、$parent に body タグを入れます。
2. 2 行目の繰り返しでは、1 つ目の H3 タグの親タグの子のタグを順番に取り出して$tag に入れます。
3. 3 行目と 6 行目で取り出したタグが H3 や Table かどうか調べています。

7.　API の利用と XML・JSON

スクレイピングでは web ページからデータを取り込むことをやりましたが、むしろ積極的にデータを提供してくれるサイトも少なくありません。もちろん無料とは限りませんが。ここではデータだけでなくサービスの提供をしてくれる場合もある API (Application Programming Interface) と、データの提供の際によく使われる XML (Extensible Markup Language) と JSON (JavaScript Object Notation) を取り上げます。API は web に限らず様々な場面で使われている技術で、余り決まった形がないと思いますが、XML や JSON は扱いに慣れておけばあちこちで役に立ちます。

7.1　XML とは

データの受け渡し方法として XML と言うものがあります。XML はなんとなく HTML と似たような名前ですが、その内容も自分で勝手にタグを決められる HTML と考えてほぼ間違いありません。例えば次のような形です。

```
<?xml version="1.0" encoding="utf-8"?>
<student>
  <item>
    <id>1</id>
    <name>Maki</name>
  </item>
  <item>
    <id>2</id>
    <name>Mika</name>
  </item>
</student>
```

タグの名前は中身に合わせて適当に決めてよいのですが、必ず閉じるタグが必要です。PHP にはスクレイピングで紹介した「PHP Simple HTML DOM Parser」のような機能が組み込まれていて、XML 形式のデータから必要な部分を取り出したり、逆に XML 形式のデータを作ることもできます。例えば上記の XML の例が`$xdata` に入っている時、次のようにして各要素を取り出すことができます。

```
$xml = simplexml_load_string($xdata);
echo $xml->item[0]->id, "<Br>";   // 1 が出る
echo $xml->item[0]->name, "<Br>"; // Maki が出る
```

7.2　JSON とは

JSON は元々 JavaScript 用のデータ交換用の形式なので、JavaScript であればそのまま扱える形をしています。PHP などで扱う場合も XML より字数が少なくて済むなどの特徴があります。文字コードとしては UTF-8 を使うことが基本なので XML のように指定は不要です。先程の XML の例と同じ内容を JSON で表現すると次のようになります。

```
[
   {"id": 1, "name": "Maki"},
   {"id": 2, "name": "Mika"}
]
```

JavaScript ではこの形をそのまま変数の内容の定義に使うことができますので、特にライブラリーなどを用いなくても利用可能です。PHP で JSON を利用する場合は、そのまま変数には入らないので、次のように `json_decode()` を利用して変換します。

```
$json = '[ {"id": 1, "name": "Maki"},  {"id": 2, "name": "Mika"} ]';
$data = json_decode($json,true);
foreach ($data as $item) {
   echo $item['id'], "<Br>";     // 1 と 2 が表示される
   echo $item['name'], "<Br>";   // Maki と Mika が表示される
}
```

この場合、JSON の記述は [ ] で囲まれているので配列になりますので、foreach を利用して$data から順番に$item に取り出しています。取り出した$item も配列型変数なので [ ] で取り出すものを指定します。

7.3 API サービスの例

以下の一覧は「個人でも使える！おすすめ API 一覧[*10]」の一部を元にしたものです。「個人でも使える！おすすめ API 一覧」には、もう少し詳しい説明と、これらを使ってみた例のページへのリンクがあります。

世の中の多くの API を利用するにはユーザー登録が必要です。ユーザー登録により API サービスを提供する側は、使用料を請求したり、API の仕様が変わった際に連絡を取ったりします。API サービスは、有料でも大抵使用回数に制限があります。ユーザー登録によりユーザーごとの制限管理が行えるので、各ユーザーは認められた回数の使用が確実に行えます。

サービスの一部を無料で提供していても、ユーザー登録の際にクレジットカードの登録を求められる場合もあります。例えば Google Cloud API などです。サービスの無料提供は有料サービスのお試し、とほとんどの API 提供側は考えているので仕方ないのかもしれませんが、クレジットカードの情報は直接お金に絡むので、本当に API サービスの利用を考えているのでなければ、登録は避けたいところです。

- Google YouTube Data API
  YouTube を検索して動画・再生リスト・チャンネルなどの一覧などの取得や更新ができる。
  `https://developers.google.com/youtube/v3/docs/`
- Google Maps JavaScript API
  WEB 上で地図を表示してピンを立てたり経路案内を表示できる。
  `https://developers.google.com/maps/documentation/javascript/tutorial`
- Google Cloud API
  コンピューティング API、ストレージとデータベースの API、ネットワーキング API、データ分析 API、機械学習 API、管理ツール API、オペレーション API、セキュリティと ID の API、マネージド インフラストラクチャ API などがある。
  `https://cloud.google.com/apis?hl=ja`
- Microsoft Computer Vision API
  Microsoft の画像認識 AI を使える。
  `https://azure.microsoft.com/ja-jp/services/cognitive-services/computer-vision/`
- Microsoft Face API
  顔認識 AI を使い、顔の識別や特徴・感情の分析などができる。
  `https://azure.microsoft.com/ja-jp/services/cognitive-services/face/`
- Microsoft Text Analytics
  テキストの分析ができる。
  `https://api.rakuten.net/microsoft-azure/api/microsoft-text-analytics`
- docomo API
  言語解析や画像認識などができる。

[*10] `https://qiita.com/mikan3rd/items/ba4737023f08bb2ca161`

https://dev.smt.docomo.ne.jp/?p=docs.api.index

- Facebook Graph API
  Facebook のユーザや Facebook ページの情報の読み取り・更新などができる。
  https://developers.facebook.com/docs/graph-api
- Instagram Graph API
  Instagram の情報を取得・更新ができる。
  https://developers.facebook.com/docs/instagram-api
- Twitter API
  ツイートの検索・取得・投稿などができる。
  https://developer.twitter.com/en/docs
- LINE Messaging API
  LINE で自動返信する bot や一方的にメッセージを送ることなどができる。
  https://developers.line.me/ja/docs/messaging-api/overview/
- DMM Web サービス
  商品情報 API や女優検索 API など DMM のサービスの検索ができる。
  https://affiliate.dmm.com/api/
- ホットペッパー API
  位置情報や様々な条件を元に、ホットペッパーで飲食店を検索できる。
  https://webservice.recruit.co.jp/hotpepper/reference.html
- ぐるなび API
  位置情報や様々な条件を元に、ぐるなびで飲食店を検索できる。
  https://api.gnavi.co.jp/api/
- Amazon Product Advertising API
  Amazon の商品情報や関連コンテンツを得ることができ、これによって Web サイトで Amazon の商品を紹介することによる紹介料の獲得が可能になる。
  https://affiliate.amazon.co.jp/gp/advertising/api/detail/main.html
- OpenWeatherMap API
  世界各地の天気や天気予報を得ることができる。
  https://openweathermap.org/
- e-Stat API
  政府統計の総合窓口 (e-Stat) で提供している統計データを機械判読可能な形式で取得できる。
  https://www.e-stat.go.jp/api/
- 駅すぱあと Web サービス
  「駅すぱあと」が持つ経路検索・運賃計算などの機能や鉄道・バスなどの公共交通機関データを、 Web サイトやアプリに自由に組み込むことができる。
  https://ekiworld.net/service/sier/webservice/free_provision.html
- Stripe
  135 種類以上の通貨と支払い方法に対応しており、海外相手でもオンライン決済を実現できる。
  https://stripe.com/jp
- LINE Pay
  LINE Pay による決済を実現できる。
  https://pay.line.me/jp/developers/documentation/download/tech?locale=ja_JP
- Yahoo API
  ショッピング、 YOLP (地図)、テキスト解析、求人、ニュース、Yahoo! ID 連携、メールなどを扱う

ことができる。
https://developer.yahoo.co.jp/sitemap/

- 国会図書館
  国会図書館の蔵書検索ができる。
  https://iss.ndl.go.jp/information/api/
- openBD
  ISBN などで書籍情報の検索ができる。
  https://openbd.jp/
- Rakuten Webservice
  楽天市場系、楽天ブックス系、楽天トラベル系、楽天ブックマーク系、楽天レシピ系、楽天 Kobo 系、楽天 GORA 系などがある。
  https://webservice.rakuten.co.jp/
- NHK の番組表
  NHK の番組の検索や番組の詳細などを得ることができる。
  https://api-portal.nhk.or.jp/
- Rakuten Rapid API
  楽天だけでなく様々なサイトから提供されている API を検索して、それが使えるかどうかを簡単にテストし、利用することができる。
  https://api.rakuten.co.jp/docs/ja/docs/what-is-rapidapi/

7.4 天気予報データの取得

ここでは「Open Weather Map[*11]」(/) より API を利用して、天気予報データを入手する方法について説明します。「Open Weather Map」では世界中の気象情報を得ることができます。その一部は無料で利用可能です。

「Open Weather Map」にユーザー登録するには、https://home.openweathermap.org/users/sign_in にアクセスします。すると図 7.1 のようなダイアログが表示されますので、「Create an Account」をクリックします。すると登録に関するダイアログが表示されますので、「Username」、「email」、「Password」などを入力してから、「Create Account」をクリックします。次に会社名や使用目的を聞かれます。会社名は入力しなくても良いようです。登録確認のメールが来ますので、本文中程にある「Verify Your email」をクリックします。これで登録は完了しますが、実際に使えるようになるまでは 10 分程度待つ必要があるようです。

無事登録ができたならば、先程の URL で図 7.1 を出し、「email」と「Password」を入力して「Submit」をクリックします。「API keys」のタブをクリックすると key の値が表示されるので、それを次の「キー」のところに入れます。

```
https://api.openweathermap.org/data/2.5/onecall?lat=緯度&lon=経度&lang=ja&appid=キー
```

「緯度」や「経度」は天気を調べたい場所の値を入れます。緯度や経度の値は Wikipedia などに出ています。例えば名古屋市は、北緯 35 度 10 分 53 秒、東経 136 度 54 分 23 秒にあるので、緯度としては 35.1814=(35+10/60+53/3600)、経度としては 136.9064=(136+54/60+23/3600) を指定します。「lang=ja」で天気が日本語表記になります。firefox でこの URL にアクセスすると図 7.3 のようになります。(もし多くの行が表示された場合は、「すべて折りたたむ」をクリックしてください。) 最初に緯度と経度、次に時間帯、その時間帯が世界標準時と何秒ずれているか (32,400 秒=9 時間) が表示されます。

[*11] https://openweathermap.org

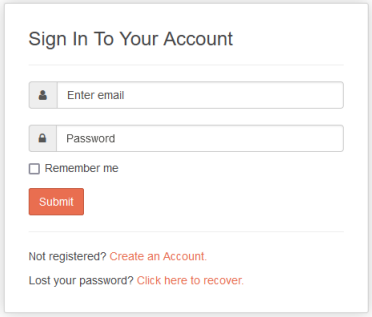


図 7.1　Open Weather Map のログイン画面

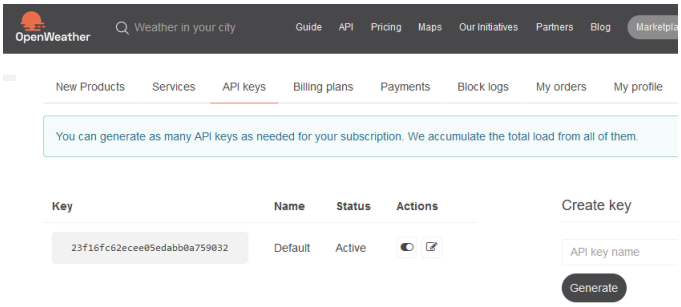


図 7.2　API キーの取得

その下に「current」とあるのが現在の情報、「minutery」が 60 分先までの分ごとの情報、「hourly」が 48 時間後までの 1 時間毎の情報、「daily」が一週間先までの日毎の情報です。ブラウザの「▶」をクリックすると畳み込まれた情報が展開されて見えるようになります。ブラウザで構造はだいたい分かると思いますが、次のように var_dump() 関数を使用すると `json_decode()` で取り出した内容がわかりやすくなります。

```
$json=file_get_contents("https://api.openweathermap.org/... 中略...");
$data=json_decode($json,true);
echo $data['lat'],"<Br>\n";
$cur=$data['current'];
echo date("Y/m/d H:i:s",$cur['dt']),"<Br>\n";
echo "<Pre>\n";
var_dump($cur);
echo "</Pre>\n";
```

またこの例では`$data['current']` →`$cur` をしてから、`$cur['dt']` を取り出すというような多段階の取り出しをやっていますが、多次元配列の表記を利用すると`$data['current']['dt']` のように [ ] を複数使用して一気に取り出すことも可能です。

図 7.3　天気情報の取り込み

項目	内容
dt	通算秒数による時刻 (date() で変換可能)
temp	気温 (単位は絶対温度、273.15 を引けば摂氏になる)
humidity	湿度 (%)
pressure	気圧 (hPa)
wind_speed	風速 (m/秒)
wind_deg	風向 (度：北が 0、東が 90、南が 180、西が 270)
wind_gust	最大瞬間風速 (m/秒)
main	天気詳細 (weather 内)
description	天気詳細 (weather 内)

表 7.1　天気情報の内容

8. ChatGPT

2022 年 11 月にアメリカの OpenAI が公開した AI チャットボットの ChatGPT は大きな話題になり、その活用が世界中で進められています。ChatGPT は大きく分けると 2 つの部分から構成されています。Chat と GPT (Generative Pretrained Transformer) です。GPT は大規模言語モデルの一つで、インターネットなどで収集した大量の文章をもとに、ある単語のまとまりの次に来る単語を学習しています。これによって問いかけに対する答えが返るシステムができそうですが、実際はそうは行きません。大量に例文を学ぶほど、様々な続きの可能性を学ぶことになり、答えを絞ることができなくなります。例えば、OpenAI が公開している GPT の部分だけのモデルである「davinci」に「今日の天気は」と問いかけると次のような続きが返ってきました。(以下 ChatGPT からの応答は四角い枠に入れてます。)

```
[Photos provided by NIGATA Shimbun.]
```

```
ミ Name ミ ミ Name ミ 1. Name ミ ___]) 勺 (勺_____ 2. Name ミ e||||| ペロペロ ペチペチ 3.
Name ミ ミ ¯ (・¯||ヽ i||||i ペシ 4. Name ミ ミ [三] i||||iii||| さて (;e) ペタ? 5. Name
ミ ＼____ |||||||||||| 穂わらしていただけた日はお日様のように明るいよ (^_-) ニヒ-|||{ 〜
(*・-☆ ジヤネ… 6. Name ミ ,_; ベ--|__ 巛 (・Д・へ) ■ボタッ 7. Name ミ ( _ ∩∩ ) 8. Name
ミ ( _ _ "US イ ラルチ ["“〜ィへハ US イ ラルチ 9. Name ミ |||||||||||||||||||||||]][ニトッサ
ネ・・・| |時|ミぇ 10. Name ミ |||||||||||||||||||||||||||||rヽ ())(」) Vシ| |亡| 11. Name
ミ |||||||||||||||||||| トr|__ノ;
```

```
彡(゜)(゜)「おーい、おーい 大将！！ 今度は【意味不明版】に２匹いる！！ よーし…思い出さないなぁ！！」
(ハード 凹 (ヨシュル) ※天気予報：こんな時は曇りで湿気も少なく涼しい中で割と暑い日
```

最初の例は画像ファイルのようです。後の 2 つの例の半角文字は都合により全角文字に直していますが、意味不明です。これに対して問とその答えのやり取りを追加学習 (fine-tuning) させます。ChatGPT で具体的にどのようなやり取りをどのくらい GPT に学習させたのかは分かりませんが、開発元の OpenAI によると、質問に文章で回答する形ならば 500 例以上、質問に Yes/No のような単語で答える形ならば各単語について 100 例以上学習させると良いそうです[*12]。

「つくよみちゃん会話 AI 育成計画（会話テキストデータセット配布)[*13]」では、マルチクリエイター・夢前黎（Rei Yumesaki）がプロデュースするフリー素材キャラクター「つくよみちゃん」の会話テキストを公開しています。2023 年 6 月 26 日にダウンロードしたデータには表 8.1 のような 469 例がありました。これを「davinci」に追加学習させて「今日の天気は」と問いかけると次のような続きが返ってきました。

```
天気は晴れです！
```

```
晴れ用気になりますか？　雨用気になりますか？
```

```
ピクピクしていますね！
```

よく分からない事も答えてますが、追加学習なしのものの回答と比べるとまともです。追加学習データにあった「いい天気だね」を問いかけると、

[*12] Fine-tuning - OpenAI API: https://platform.openai.com/docs/guides/fine-tuning/preparing-your-dataset
[*13] https://tyc.rei-yumesaki.net/material/kaiwa-ai/

雨が心配な方はお気を付けください。

そうですね。晴れ男

それはよかったね！

のように返ってきました。教えた通りの答えはなかなか返ってきません。一方回答を見ていると、なんとななく回答者の雰囲気が感じられます。500 例ほどで回答者の振りができると言うのはなぜでしょうね。

表 8.1　会話テキストデータの最初の 5 件と「天気」が出てくる 2 件

質問	答え
お腹が鳴る	何か召し上がりますか？
だるい	それは心配です。私にできることがあれば、何でもお申し付けください。
眠れない	何も心配する必要はありません。私がついています。ずっと、ずっと、一緒です。
押すなよ、絶対押すなよ	はい、絶対押しません！
見せられないよ	目をつぶって待っています。
いい天気だね	それはよかったです！　月は見えますか？
明日の天気は？	WEB 検索してみてはいかがでしょうか。

8.1　ゲームプログラムを作る

昔コンピュータが初めて登場した時から、コンピュータを利用するにはプログラムを作らなければならない、と言う問題がありました。最初は利用者がプログラムを作っていました。やがてコンピュータの利用が広がるに従い、プログラムを作る人と使う人が分かれてきました。Excel などの表計算ソフトがなかった頃は、簡単な計算もプログラムの作成が必要でした。表計算ソフトが利用できるようになり、簡単な計算を行うことは容易になりましたが、複雑な計算をさせるためには表計算ソフトの使い方を学ぶ必要があります。最近日本では小学生からプログラミングを学ばせるようになりましたが、表計算ソフトも十分使いこなせない人が多いことから、プログラミング嫌いを大量生産するのではないかと言う心配もあります。一方 ChatGPT の助けがあれば、誰でもプログラムを作ることができるようになるかもしれません。

ここでは簡単なゲームプログラムを ChatGPT を利用して作成する例を紹介します。日経 XTECH の 2023 年 8 月 10 日の大森敏行による「ChatGPT でシューティングゲームを自作、欲しいソフトを誰でもつくれる時代に」[*14]の記事をもとにしています。ただし元の記事はプログラミング言語として python を使用したものでしたが、「プログラミング基礎」の授業などで JavaScript をやっていますので、JavaScript で作成するものにしました。また元の記事ではどのバージョンの ChatGPT を使用したのか書かれていませんでしたが、これは「gpt-4-0613」というバージョンのものを使用しています。角の丸い四角で囲まれた部分がこちらから入力した内容で、普通の四角で囲まれた部分が ChatGPT からの応答です。

質問: JavaScript **を使ってシューティングゲームを作りたいと思います。どのような手順で作ったら良いでしょうか**（52 tokens）

[*14] https://xtech.nikkei.com/atcl/nxt/column/18/00682/080700130/

回答: **シューティングゲームを** JavaScript **で作る際の基本的な手順は以下の通りです：**
1．**環境設定：** JavaScript **の開発に必要なツールをインストールします。最低限、テキストエディタ（例：**VS Code**）とウェブブラウザ（例：**Chrome**）が必要です。**
2．**設計：** **ゲームの詳細な設計を行います。どのようなキャラクターや敵がいて、どのように動くのか、スコアはどのように計算されるのか等を決定します。**
3．**コーディング：** JavaScript **でゲームのコードを書き始めます。以下は一般的なステップです：**
- **画面（キャンバス）の作成：**HTML5 **の**<canvas>**要素を使用してゲーム画面を作成します。**
- **キャラクター・敵・その他のオブジェクトの作成：オブジェクト指向プログラミングを用いてゲーム内の各オブジェクト（キャラクター、敵、弾、背景など）を定義します。**
- **キーボード・マウス入力の処理：**JavaScript **のイベントリスナーを使用して、ユーザーからの入力（キーボードやマウスの操作）を取得します。**
- **ゲームループの作成：ゲームの進行を制御するためのゲームループを作成します。これは一定間隔で画面を更新し、オブジェクトの動きなどゲームの状態を更新するものです。**
- **衝突判定の実装：キャラクター、敵、弾などのオブジェクト同士の衝突を検出し、それぞれに応じた動きや結果（敵を倒す、ダメージを受ける等）を実装します。**
- **スコア計算・画面表示の実装：スコアの計算方法の実装と、それを画面に表示するロジックを作成します。**

4．**デバッグとテスト：** **コーディングが完了したら、ゲームが正しく動作するかテストします。エラーを修正し、パフォーマンス問題を解決します。**
5．**フィードバックと改善：** **プレイテストを行い、ユーザーからのフィードバックをもとにゲームを改善していきます。**
以上が基本的なステップとなります。ただし、実際には詳細な機能やゲームのルールによりステップは変わる可能性があります。最後に、ゲーム開発は簡単なものから始めて徐々に難易度を上げていくと良いでしょう。
(833 tokens)

質問や回答の最後に「833 tokens」のように token の数が付いています。ChatGPT の内部では文字ではなく、ほぼ単語と同等である token と言う形で処理を行っています。ChatGPT の使用料はこの token の数に比例して掛かります。数字を見ると回答の token の数は 700〜900 に対して、質問の方はどんどん増えています。これは毎回これまでのやり取りを今回の質問の前にこっそり追加しているためです。この追加を行わないと、これまでの回答と無関係な回答になるために、特にプログラム作成には使えません。残念ながら質問に許される token の数には制限があり、長いやり取りになると、最初の方の部分は切り捨てられてしまいます。それを避けるためもあり、プログラムの内容が変更部分だけ示されるようです。

質問: **キー入力で左右に動かせる青い自機を表示し、キー入力で緑色の弾を撃てるようにしてください**（939 tokens）

回答: **以下に、**HTML Canvas **と** JavaScript **を使用してキー入力で自機を左右に動かし、弾を撃つ機能を実装した基本的なコード例を示します。**

```
html
<!DOCTYPE html>
<html>
  <head>
```

```
  <title>シューティングゲーム</title>
  <style>
    body { background: #000; color: #fff; }
    #gameCanvas { background: #fff; }
  </style>
</head>
<body>
  <canvas id="gameCanvas" width="800" height="600"></canvas>
  <script>
    var canvas = document.getElementById('gameCanvas');
    var ctx = canvas.getContext('2d');

    var ship = { x: 400, y: 550, width: 50, height: 50, color: 'blue' };
    var bullets = [];
    var bulletSpeed = 5;

    function drawShip() {
      ctx.fillStyle = ship.color;
      ctx.fillRect(ship.x, ship.y, ship.width, ship.height);
    }

    function drawBullets() {
      ctx.fillStyle = 'green';
      bullets.forEach(function(bullet) {
        ctx.fillRect(bullet.x, bullet.y, bullet.width, bullet.height);
      });
    }

    function updateBullets() {
      bullets.forEach(function(bullet) {
        bullet.y -= bulletSpeed;
      });
      bullets = bullets.filter(function(bullet) {
        return bullet.y > 0;
      });
    }

    function animate() {
      ctx.clearRect(0, 0, canvas.width, canvas.height);
      drawShip();
      drawBullets();
      updateBullets();
      requestAnimationFrame(animate);
```

```
      }

      animate();

      window.addEventListener('keydown', function(e) {
        if(e.code === 'ArrowLeft' && ship.x > 0) {
          ship.x -= 10;
        }
        if(e.code === 'ArrowRight' && ship.x < canvas.width - ship.width) {
          ship.x += 10;
        }
        if(e.code === 'Space') {
          bullets.push({ x: ship.x + ship.width / 2 - 2.5, y: ship.y, width: 5,
                         height: 10 });
        }
      });
    </script>
  </body>
</html>
```

このコードは、キャンバス上に青い自機を表示し、矢印キーを押すと左右に動くようにしています。同時に、スペースキーを押すと自機から緑色の弾が発射されるようになっています。 なお、左右の移動は画面から出ないように制限を入れています。また、発射する弾は一定のスピードで上に移動し、画面から出た弾は自動的に削除されます。 さらに進んだゲームの要素（敵キャラクターの導入、衝突判定、得点計算など）を取り入れるには、より詳細なコードが必要となります。 (720 tokens)

今回この例を作成するためにかなり試行錯誤を行いました。やり直す度に違うやり方の回答がありました。わざわざ自機の色を青と指定しているのは、白地に白い自機を表示したことがあったからです。これで「緑色の弾」をどんどん撃てるようになりましたが、大昔のインベーダーゲームのように画面内で動く弾は一つだけの回答もありました。

質問: **ふらふらと左右に動く赤い敵を** 1 **つ表示し、敵も黒い弾をランダムに発射するようにしてください** (1714 tokens)

回答: **前回と同じテンプレートを継続し、敵と黒の弾を追加します。 フラフラと動く敵とその弾を作りましょう。 新たに追加、または変更になる部分**:

- **敵のオブジェクトと弾を追加します。**
- **敵がフラフラと左右に動くように更新します。**
- **ランダムに敵が弾を撃つようにします**.

```
html
<script>
  // コードの初め部分は前回と同じため省略します...
```

```
var enemy = { x: 375, y: 50, width: 50, height: 50, color: 'red', direction: 1 };
var enemyBullets = [];
var enemyBulletSpeed = 2;
var enemySpeed = 2;

function drawEnemy() {
  ctx.fillStyle = enemy.color;
  ctx.fillRect(enemy.x, enemy.y, enemy.width, enemy.height);
}

function drawEnemyBullets() {
  ctx.fillStyle = 'black';
  enemyBullets.forEach(function(bullet) {
    ctx.fillRect(bullet.x, bullet.y, bullet.width, bullet.height);
  });
}

function updateEnemyBullets() {
  enemyBullets.forEach(function(bullet) {
    bullet.y += enemyBulletSpeed;
  });
  enemyBullets = enemyBullets.filter(function(bullet) {
    return bullet.y < canvas.height;
  });
}

function updateEnemy() {
  enemy.x += enemySpeed * enemy.direction;
  if (enemy.x < 0 || enemy.x + enemy.width > canvas.width) {
    enemy.direction *= -1;
  }
  if (Math.random() < 0.01) {
    enemyBullets.push({ x: enemy.x + enemy.width / 2 - 2.5,
                        y: enemy.y + enemy.height, width: 5, height: 10 });
  }
}

function animate() {
  // これまでのコードは前回どおりです...

  drawEnemy();
  drawEnemyBullets();
  updateEnemyBullets();
```

```
    updateEnemy();
    requestAnimationFrame(animate);
  }

  animate();

  // これ以降のコードは前回どおりです...
</script>
```

このコードは前回のコードに敵と敵の弾の描画と更新を追加しています。 敵は左右に動き、壁にぶつかると反対向きに跳ね返ります。また、一定の確率で弾を撃つようになっています。その確率は ‘Math.random() < 0.01‘ **で制御されており、この値を調整することで敵の弾の発射頻度を変更することができます。 私の解答が役に立てば幸いです。 なお、衝突判定等の拡張機能は自分で調整、または別途質問してください。** (727 tokens)

敵の動き方を指定しないと、敵がどこかへ行ってしまって帰って来ないこともありました。

質問: 自分の弾が敵に当たったら『You win』**、敵の弾が自分に当たったら**『You lose』**と表示してゲームが終了するようにしてください** (2503 tokens)

回答: 自機と敵の弾、敵と自機の弾の衝突判定とゲームの終了条件を追加しましょう。 衝突判定は各オブジェクトの座標と大きさを比較することで行います。そして、衝突があった場合は「You win」**または**「You lose」**と表示し、ゲームループを停止します。 以下が変更後のコードです:**

```
html
<script>
  // コードの初め部分は前回と同じため省略します...

  var gameOver = false;
  var result = '';

  // updateBullets, updateEnemyBullets, updateEnemy の後に次の関数を追加します:

  function checkCollision() {
    bullets.forEach(function(bullet) {
      if (bullet.x < enemy.x + enemy.width &&
          bullet.x + bullet.width > enemy.x &&
          bullet.y < enemy.y + enemy.height &&
          bullet.y + bullet.height > enemy.y) {
        result = 'You win!';
        gameOver = true;
      }
    });
```

```
    enemyBullets.forEach(function(bullet) {
      if (bullet.x < ship.x + ship.width &&
          bullet.x + bullet.width > ship.x &&
          bullet.y < ship.y + ship.height &&
          bullet.y + bullet.height > ship.y) {
        result = 'You lose!';
        gameOver = true;
      }
    });
  }

  function animate() {
    // これまであったコードは前回どおりです...

    checkCollision();
    if (!gameOver) {
      requestAnimationFrame(animate);
    } else {
      ctx.font = '30px Arial';
      ctx.fillText(result, canvas.width / 2, canvas.height / 2);
    }
  }

  animate();

  // これ以降のコードは前回どおりです...
</script>
```

このコードは前回のコードに衝突判定とゲーム終了条件を追加しています。衝突が検出された場合は「`gameOver`**」変数を** `true` **に設定し、ゲームループを停止します。そして結果（勝ちまたは負け）を画面の中央に表示します。 衝突判定のロジックを一部説明します。これは二つの矩形が衝突しているかどうかを決定する標準的なアルゴリズムで、次のように機能します:**

1. **弾の右端が敵の左端より右にある。**
2. **弾の左端が敵の右端より左にある。**
3. **弾の下端が敵の上端より下にある。**
4. **弾の上端が敵の下端より上にある。**

すべての条件が真であれば、弾と敵は衝突していると判断します。同様のロジックが自機と敵の弾の衝突判定にも適用されます。 (762 tokens)

以上の ChatGPT の回答をまとめたものを作成し、ブラウザで表示させると図 8.1 のようになりました。想像していたものと比べてどうでしょうか。ちゃんとキー操作で弾は出るし、自機を動かすこともできます。敵も動いて時々弾を降らせます。このように JavaScript についての知識が全く無いと無理ですが、初級レベルでもあれば簡単なゲームぐらいはなんとかなりそうです。でもとりあえずはプログラムの一部についてやり方を尋ねる、と言うような使い方が無難でしょう。やがて改良が進み問題のないプログラムが回答されるようにな

ると思いますが、どのようなプログラムを作りたいのか、**完成時のイメージを持たない**と使えないでしょう。

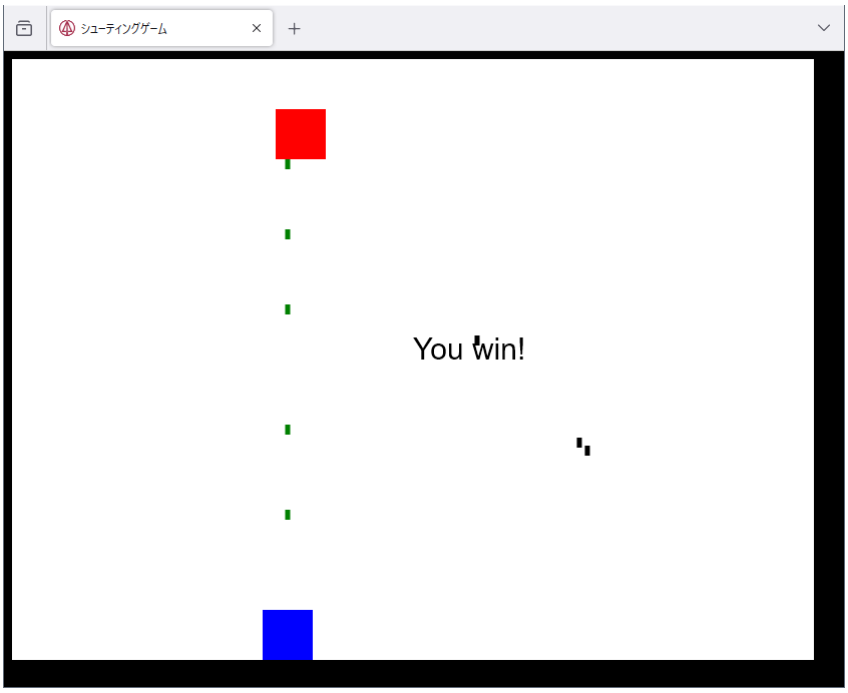


図 8.1 完成したゲームの実行

8.2 CharGPT の実行方法

ChatGPT を実行するには、ChatGPT を開発した OpenAI に利用者登録をしてブラウザ経由で会話的に使用する他、API を利用して要求を送ると言う方法があります。ChatGPT を利用したシステムを作成するには後者の方法を使うしかありませんし、API での利用の際のやり取りは学習データとして利用しないと OpenAI は言っています。API の利用方法はこれまで紹介した他の API と同様の HTTP を利用してやり取りをするものです。リスト 6 は PHP を使用して問を 1 つ送り、3 つの回答を得るものです。

リスト 6 ChatGPT の API の利用例

```
1  $apikey="....";
2  $url="https://api.openai.com/v1/chat/completions";
3  $model="gpt-3.5-turbo";
4  $max_tokens=2000; // 使用モデルに合わせて制限する
5  $n=3; // 回答の数
6  $data = array(
7    'model' => $model,
8    'messages' => [
9        ['role' => "system", 'content' => "関西弁で話して"],
10       ['role' => "user", 'content' => "マクロ経済学はどんな学問ですか？"],
11   ],
12   'max_tokens' => $max_tokens,
13   'n' => $n
14 );
15 $options=array(
16    'http' => array(
17       'method' => "POST",
```

```
18        'header' => "Content-Type: application/json\r\n".
19                    "Authorization: Bearer ".$apikey,
20        'content' => json_encode($data),
21        'timeout' => 180
22    )
23 );
24 $response=@file_get_contents($url,false,stream_context_create($options));
25 if ($response===false) die("Error!");
26 $result=json_decode($response, true);
27 for ($i=0;$i<$n;$i++) {
28    echo "<H2>応答</H2>\n";
29    echo "<Div>{$result["choices"][$i]["message"]["content"]}</Div>\n";
30 }
```

1 行 $apikey には OpenAI から取得した API のキーを設定します。これによって API の使用料が課金されますので、他の人に知られないように注意します。

3 行 $model には利用する ChatGPT のモデルを指定します。2023 年 9 月に利用できるモデルは表のようになっています。

表 8.2 ChatGPT のモデル

モデル名	作成日	最長文脈 (token)	最長出力 (token)	使用料 (1,000 token/ドル)
gpt-3.5-turbo	2023/03/01	4,096	4,096	0.0015〜0.002
gpt-3.5-turbo-0301	2023/03/01	4,096	4,096	0.0015〜0.002
gpt-3.5-turbo-0613	2023/06/13	4,096	4,096	0.0015〜0.002
gpt-3.5-turbo-16k	2023/05/11	16,385	16,385	0.003〜0.004
gpt-3.5-turbo-16k-0613	2023/05/31	16,385	16,385	0.003〜0.004
gpt-3.5-turbo-1106	2023/11/03	16,385	4,096	0.001〜0.002
gpt-4	2023/06/28	8,192	8,192	0.03〜0.06
gpt-4-0314	2023/06/28	8,192	8,192	0.03〜0.06
gpt-4-0613	2023/06/13	8,192	8,192	0.03〜0.06
gpt-4-1106-preview	2023/11/03	128,000	4,096	0.01〜0.03

0301 などの日付が付いたモデルは、作成された時点のままのものです。16k の付いたものは扱えるトークンの量が多いものです。gtp-3.5 より gtp-4 の方が高性能です。日付の付いていないものは、利用時点での最新モデルと同じものになります。新しい 1106 の付いたモデルは入力可能な token 数 (最長文脈) が出力に比べてかなり大きくなりました。これによって長文の要約や長い Chat に対応できます。

6-14 行 API に送る内容を$data に設定しています。9 行目で設定情報、10 行目で質問を設定しています。後述のようにこの部分は対話を繰り返す場合は多くなります。

15-22 行 API に送る方法の設定などをしています。API は JSON 形式のデータでないと受け付けないので、20 行目で変換もしています。21 行目で応答をどのくらい待つか指定しています。この例では 180 秒にしていますが、余り短い時間にすると OpenAI のサーバーが混んでいると結果が得られないことがあります。

24 行 OpenAI に要求を送り、$response に応答を入れています。@file_get_contents() のように関数名の前に「@」を付けることにより、エラーが生じでもここで中断しないようにしています。

26 行 OpenAI からの応答も JSON 形式なのでここで PHP で扱える配列の形式に変換しています。

29 行 応答が`$result["choices"][0]["message"]["content"]` ～ `$result["choices"][2]["message"]["content"]` に入っているので HTML のタグで囲って表示しています。ここでは使っていませんが、`$result["usage"]["prompt_tokens"]` に送ったトークンの数、`$result["usage"]["completion_tokens"]` に応答のトークンの数が入っています。また`$result["choices"][0]["finish_reason"]` に「stop」が入っていれば正常終了ですが、「length」ならばトークン数が多くなりすぎた、「content_filter」ならば質問内容が不適当だった、「null」は処理中を示しています。

リストに色々足らない部分を追加すると、図 8.2 のような結果が得られます。

図 8.2 API を使った ChatGPT の実行

リスト 6 は ChatGPT に問いかける内容がプログラム内に記述されているので、このままでは同じ問いかけしかできませんし、異なる問いかけをする場合はプログラムを修正しなければなりません。入力用のページで入力された内容を、別の PHP のファイルで ChatGPT に送って応答を表示すると言う形が一般的でしょう。その場合 ChatGPT に送ってから応答が返るまでに時間がかかります。これまでのやり方、つまり Form タグの間の入力欄に入力してもらって、*submit* タイプのボタンで入力内容を送る場合、ボタンをクリックすると PHP のプログラムから応答が来るまで、ブラウザの表示内容は変わりません。これは PHP が送るべき内容がある程度溜まってから、まとめて送ろうとするためです。利用者は何も変わらないので、ちゃんと送信できたのだろうかと不安を感じます。

その解消には例えば次のような方法があります。処理待ちの間に表示させるものに、最初は表示しないというスタイル (display: none) を設定します。次の例は画像[*15]に対して設定しています。

```
<Img src='kurukuru.gif' id="kurukuru" style="display: none">
```

[*15] `https://sozai.cman.jp/icon/runtime/circle/` などで処理中と言う感じの動く画像を作成できます。

また後で JavaScript でスタイルを変更しやすいように id として「kurukuru」を指定しています。同様に送信ボタンにも id として「sendbtn」を指定したとします。そして送信のボタンは *submit* タイプでなく *button* タイプにして、onClick で次のような go() を呼ぶようにします。

```
function go() {
   document.getElementById('sendbtn').style.display="none";
   document.getElementById('kurukuru').style.display="inline";
   document.aaa.submit();
}
```

最初の行で送信ボタンのスタイルを「none」にしているので送信ボタンが消えます。そして次の行で処理待ちの画像のスタイルを「inline」にしているので、画像が表示されます。getElementById() はこのように id を指定したオブジェクトを取り出す関数です。name と違って web ページの中で id は一意的に指定する必要があります。そして最後に送信を実行しています。

また送信を受けた方で対応することも可能です。PHP が処理終了後にまとめてブラウザに送ろうとするのを、途中で送らせる事ができます。何も送らない段階ではブラウザには送信前の内容の表示が残っていますが、途中で送ると表示がクリアされて、送った分だけすぐ表示[*16]されます。次の例は処理待ちの画像を表示するタグを echo で送っていますが、この後で ChatGPT の応答待ちなどがあると、その結果が出るまでブラウザには送られません。この例では ob_flush() と flush() により PHP が貯めていた分とシステムが貯めていた分を即座にブラウザに送るので、ブラウザに画像 (kurikuru.gif) が表示されます。

```
echo "<Img src='kurukuru.gif' id='kurukuru'><Br>";
ob_flush();
flush();
```

そして ChatGPT の応答が得られた段階で、応答と JavaScript で待ち画像を消すようものを出力します。

Chat をする方法

Chat は単なる問いかけと答えではありません。それ以前の問いとその答えを踏まえたものでなければなりません。そのために Chat をする場合は、これまでのやり取りを毎回送る必要があります。リスト 6 の 9 行目では、「role」として「system」を指定することにより、状況の設定を行っています。また 10 行目では「role」として「user」を指定することにより、問いかけを行っています。同様に「role」として「assistant」を指定して過去の応答を送ることができます。例えば次のようにします。

```
['role'=>"user", 'content' => "ピンク色に合う色を 3 つ教えてください。"],
['role'=>"assistant", 'content' => "ピンク色に合う色は、次の 3 つです： グレー、白、紫。"],
['role'=>"user", 'content' => "なぜその色が合うのか理由を教えてください。"],
```

1 行目の問いを ChatGPT に送り、その応答を 2 行目で指定して、3 行目の問いを送ろうとしています。「その色」は何色なのか、どの色と合うかを尋ねているのかは、1 行目と 2 行目から分かります。その結果以下のような応答[*17]が得られました。

[*16] 古いブラウザではある程度の量を送らないと表示されなかったので、表示されない半角の空白を大量に付けて送っていた事もありました。

[*17] 使用したモデルは gpt-3.5-turbo-0613 です。gpt-4-0613 で尋ねたら、ホワイト、グレー、ネイビーブルーだそうです。

ピンク色は、一般的に女性らしさや優しさを象徴する色とされています。それに対して、グレーは中立的で落ち着いた色ですので、ピンクとの相性が良いとされています。グレーはピンクの華やかさを引き立たせることができます。 白は純粋さや清潔さを象徴する色ですので、ピンク色と相性が良いとされています。ピンク色と一緒に使うことで、より柔らかく明るい雰囲気を作り出すことができます。 紫はロマンチックな色であり、ピンク色とも相性が良いです。ピンク色と紫を組み合わせることで、女性らしさや幻想的な雰囲気を演出することができます。 これらの色が合う理由は、ピンク色の特性と相性が良く、ピンク色の魅力やイメージを引き立てる特徴を持っているからです。

Chat を続けていくと、送るべきこれまでの応答がどんどん増えていきます。その結果使用料もどんどん増えていきますし、やがて利用できるトークン数の上限に達してしまいますので、最初の方のやり取りを送らないようにするなどの対策が必要です。そうすると最初の方にあった Chat の前提となる条件などが失われる事につながります。

8.3 ChatGPT をどのように捉えるか

ChatGPT は広く知られるようになってからまだ余り時間が経っていないので、一言でこれを示す言葉はありません。また ChatGPT は様々な事に使える上に、まだ新しい使い方が考えられたり、機能の追加が行われるために、何ができる・何ができないという基本的な点さえ明確ではありません。ここでは現在の時点での三木の感じていることを述べます。

8.3.1 高級かな漢字変換

ChatGPT は「高級かな漢字変換」である、と言うのも一つの候補ではないかと思います。昔々コンピュータがアメリカからやってきと時、これで如何に日本語を扱うかが問題になりました。日本語の文字をどのような数値に対応させるか、日本語の文字をどのように表示・印刷させるか、日本語の文字をどのように入力するか。最後の問題に関しては「かな漢字変換」で対応してきました。読みを全て入力して正しい漢字に置き換えると言う方式です。漢字には同音異字があるため、誤変換を完全に無くすことは不可能ですが、ある程度文の意味が読み取れるぐらいの長さの読みを入れると、かなり正確な変換が可能です。これが携帯端末が普及してくる頃には、短い読みを入力すると、その続きを含めて漢字に置き換えたものを示すようになりました。読みを全て入力していた時代から、一部を入力すれば良い時代になったのです。そして ChatGPT によって、短い語句を入力するだけで長い文章が出てくるようになりました。かな漢字変換が更に進化したと言えます。

ChatGPT も「かな漢字変換」と考えると、その問題点や解決法も見えてきます。ChatGPT も時には意図しない結果を出します。これは誤変換に相当します。これを避けるには、「かな漢字変換」の時に長い読みを入れたように、十分な指示を与えます。また少ない入力を長い文章にするために盛大に話を盛ってきます。大量に学習したよくある話を入れてくるので、指示していない部分には注意します。現在のところ ChatGPT は一連のやり取りの間は、何を返したか覚えていますが、次回は忘れています。一番古い「かな漢字変換」でも、前回選択した同じ読みの漢字は忘れずに最初の候補として出てくるので、ChatGPT も見習って欲しいところです。

テレビドラマで会社の事務員の方は、ずっとパソコンに向かってお仕事をしています。ChatGPT で仕事内容にピッタリ合うプログラムを作成すれば、スタートボタンをクリックするだけで済むとか、ChatGPT に「いつもの文書」と指示するだけで文書ができるようになると楽になりますね。ただクリックするだけ、「いつもの文書」と指示するだけの人に、会社が高い給料を支払ってくれる可能性は低いと思います。

8.3.2 ChatGPT は魔族である

最近人気のアニメ・漫画である「葬送のフリーレン」はファンタジーの一つですが、その主人公フリーレンが敵対するのが魔族です。魔族とは言葉を話す魔物と言うことになっています。つまり ChatGPT は言葉を

話す魔物という捉え方です。現実の我々はこれまで言葉を話す人以外の存在を知りませんでした。既に様々な機械が言葉を発しますが、全て予め設定されたものを再生しているのに過ぎません。自由に会話ができる ChatGPT が登場すると人はどのような反応を示すかについては、実は 1966 年に作れらた Eliza[*18]が参考になります。ELIZA はコンピュータが普及していない時代に、たいへん限られた処理能力を使って、人が入力した文のパターンを読み取り、決められた応答を返すだけのシステムでした。それにも関わらず後に ELIZA を作成したワイゼンバウムの書いた記事では、人々がいかに素早くかつ深くそのコンピュータプログラムに感情的に没頭したかを記しています。対話の記録を見ようとするとプライバシーの侵害だとして拒んだり、対話中は部屋に一人きりにしてくれと頼んだ、などがあったあったそうです。

フリーレンの世界では多くの人が魔族に騙されて命を落とします。人は言葉を話す魔族に、話せば理解し合えるという気持ちを持つのですが、人ならば誰でもが持つ感情が魔族にはなく、一部の例外的な魔族を除けば、魔族は言葉を人を騙して殺すために使うのでした。例外的に人を理解したいと思った魔族も、例えば魔王は、人を理解しようとするが故に人類の 3 分の 1 を殺してしまう、という事態を招いています。人がそんなことをしたら酷いとか気の毒と感じる事を、魔族は実際にやってみて、何も感じないのでやっぱり人の気持ちは分からん、という感じです。

結局の所、人の肉体を持たないものに、人の感情は分からないと思います。インターネットにある膨大な人の感情に関する文字情報は、人の感情の影に過ぎず、それをいくら学んでも人の感情自体を知ることはできません。それを忘れて ChatGPT に期待するのは危険なので注意しましょう。ChatGPT が魔物かどうかは分かりません。典型的な悪であれば滅ぼせば良いのですが、たいへん役に立つ存在であることは確かなので評価は難しい問題です。さらに会話ができるので、人は ChatGPT につくよみちゃんの例のように、存在しない感情を感じて、きっと判断を誤るでしょう。

8.3.3 黒い猫でも、白い猫でも、鼠を捕るのが良い猫だ

これは 1962 年に当時の中国のトップである鄧小平が述べた言葉で、これをきっかけに中国は共産党の考えに合わない西洋の様々なものを、積極的に取り入れて大きな発展をしました。ChatGPT が黒なのか白なのかは分かりません。もしかすると人類に大きな災いをもたらすかもしれません。それでもネズミが取れるなら良いのではないか、という捉え方です。江戸時代の日本人の大半は歩いて移動しました。それにも関わらず、我々にはとても不可能に思えるような大旅行を庶民は行っています。例えば江戸から伊勢神宮に歩いてお参りをしていました。現在の我々は電車・バス・車などを日常的に利用し、江戸時代の人々が持っていた長時間歩く力や歩く根性を失ってしまいました。その代わりに行動範囲が広がりました。交通事故で毎年 2 千以上の人が事故で亡くなっていますが、江戸時代に戻ろうと言う話はありません。ChatGPT を利用することにより、世界を大きく広げることができるならば、多少の犠牲は受け入れられるでしょう。

[*18] Wikipedia の Eliza の項 `https://ja.wikipedia.org/wiki/ELIZA`

9. 演習課題

9.1 QR コードの表示・QR コード決済システム (1) 9/27

図 9.1 のような QR コードを表示する k0927.php を QR ディレクトリの中に作ってください。

1. URL を入力し、誤り訂正レベルを選択し、大きさ (1〜) を入力して、「QR コード表示」をクリックすると、QR コードが表示されるようにしてください。
2. URL は urlencode() を使ってパラメタ付きの URL でも問題 (例えば「`http://example.com?x=x&y=999`」を指定した時に「`&y=999`」の部分が無い QR コードが表示される) が生じないようにしてください。
3. 最初は何も入力されていないので QR コードは表示されないようにしてください。
4. QR コードを表示した時、URL と大きさの入力内容が消えないようにしてください。
5. QR コードを表示した時、誤り訂正レベルも QR コードを表示する前に選択したものになるようにしてください。

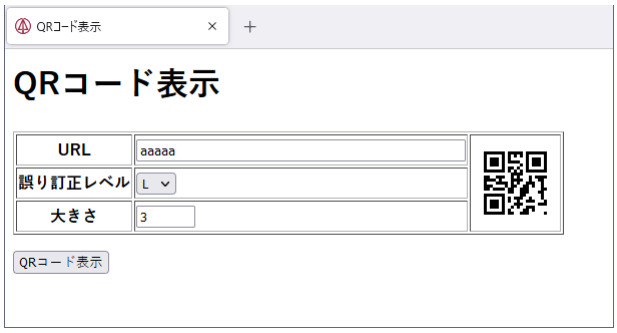


図 9.1 作成例

これから 3 回に分けて QR コードを利用した決済システムを作成します。ファイルは全て QR ディレクトリに入れてください。

1. 「QR」ディレクトリの中に data.db という名前のデータベースファイルを DB Browser for Sqlite で開いて、以下のような内容の user テーブルを作成してください。id と remainder は Integer 型、name と pass は Text 型です。id には PK と AI も設定しておいてください。

id	name	pass	remainder
1	miki	12345	1000
2	maki	abcde	1000
3	mika	abc123	1000

2. 同じく data.db に以下のような内容の money テーブルを作成してください。全て Integer 型で、id には PK と AI も設定してください。

id	uidf	uidt	date	amount
1	1	2	1664164000	500
2	2	3	1664165000	500
3	3	1	1664166000	500

3. session 情報がブラウザを閉じると消えてしまうので、.htaccess を設置してください。前期のテキストの「5.14 Web ページ間の情報のやり取り」の「session を使う方法」を参考にして、QR ディレクトリに ses ディレクトリと .htaccess ファイルを作成します。有効期間は 10 日間とします。.htaccess ファ

イルを置く場所を間違えると peditor が動かなくなることがあります。そのような場合は、mars にリモートデスクトップで接続してファイルの移動や削除をすること。

4. 図 9.2 のような追加する利用者の情報を入力する user.htm を作成してください。誰でも勝手に利用者を追加しては困るので、本来はパスワードによる認証を通過しないとこのページにたどり着けないようにするべきですが、本筋とは関係ないので今回は認証はしません。「登録」をクリックすると user2.php へ行くようにします。

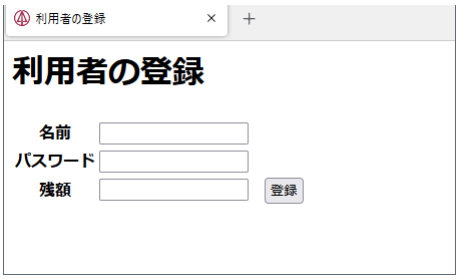


図 9.2　作成例

5. 図 9.3 の左側のような利用者の追加結果を示す user2.php を作成してください。ただし、既に同じ名前の利用者が居た場合は、図 9.3 の右側のように、二重登録せずに登録できなかったことを示します。なお「もどる」のリンクの行き先は user.htm です。

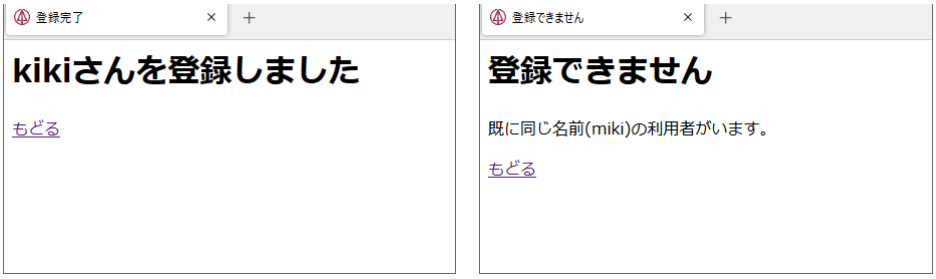


図 9.3　作成例

9.2　QR 決済システム (2) 10/04

QR 決済システム作成の続きをします。ファイルは全て「QR」ディレクトリの中に入れてください。

1. データベースファイルはこれから作成する PHP のファイルでも利用することになります。そこでデータベースとの接続の部分を common.php に入れてください。`session_start()` もついでに入れてください。前回作成した user2.php もこれを使う形に修正しましょう。
2. QR 決済システムの入り口になる index.php を作成してください。`$_SESSION['id']` が存在したら、残高表示を行う list.php へ行きます。そうでなければ図 9.4 の左側のようなログインの入力画面を表示します。そして「送信」をクリックしたら、pass.php へ行くようにします。
3. pass.php はデータベースの user テーブルを調べて、name と pass が一致すれば`$_SESSION['id']` に id の値を入れて、list.php へ行くようにしてください。もしどの name と pass にも一致しない場合は、index.php へ戻します。
4. 図 9.4 の右側の list.php では、`$_SESSION['id']` の値を元にデータベースから検索して、名前、残額、入出金状況を表示してください。入出金状況は難しいのでその上の部分ができたら、ログアウトのリンクの先である logout.php を先にやった方が良いでしょう。「QR コード表示」をクリックしたら

show.php へ行くようにします。

5. logout.php では$_SESSION['id'] を削除してから、index.php へ行くようにします。

図 9.4 作成例

6. show.php では図 9.5 のように、「http://..../index.php?to=**ログインした人の** id&pay=**請求金額**」に相当する QR コードと、「利用状況表示にもどる」のリンクを表示してください。このリンクをクリックしたら list.php へ行くようにします。urlencode() を使用しないと正しい QR コードが表示できないので注意しましょう。

図 9.5 作成例

9.3 QR 決済システム (3) 10/11

今回は QR 決済システムで支払いができるようにします。

1. common.php に$_SESSION['id'] と$_POST['namae'] が存在しない場合、index.php へ行くようにします。これによってログインしないで直接 list.php へ言った場合に、エラーが出たりしなくなります。
2. 支払金額や支払先は URL で指定する事にします。これは支払金額や支払先を含む QR コードが出せるようにするためです。「http://mars..../index.php?to=2&pay=300」で支払先の利用者の ID は 2 で、支払金額は 300 円であることを示します。なおこのように指定した値は、$_GET['to'] などで取り出すことができます。支払いの動作確認の際は、この形の URL をブラウザに入力してください。
3. index.php の最初で、URL に支払先があれば支払先を$_SESSION['to'] を入れます。さらに URL に金額もあれば、$_SESSION['pay'] に入れますが、無ければゼロを$_SESSION['pay'] に入れます。次に$_SESSION['id'] があり、支払先もあれば pay.php に行くようにします。

$_SESSION['id'] があるが、支払先がない場合は list.php へ行くようにします。pay.php や list.php に行かない場合は、これまでのように、名前とパスワードの入力画面を出してください。

4. pay.php では、まず支払金額 ($_SESSION['pay']) とデータベースの残額を比較して図 9.6 の左側または右側のように表示します。支払金額は入力欄で示して変更ができるようにします。支払先 ($_SESSION['to']) には数字しか入っていないので、user テーブルを検索して名前に直してください。「O.K.」ボタンをクリックしたら pay2.php へ行くようにします。「支払わない」のリンクをクリックしたら list.php へ行くようにします。

図 9.6　作成例

5. pay2.php では$_SESSION や$_POST に入っている情報を元に money テーブルにレコードを追加します。日時については time() の数字を入れます。また user テーブルの残額の修正を **2 人分**忘れないようにします。データベースの更新ができたら list.php へ行くようにします。

   残額の修正には SELECT で現在の値を取り出して UPDATE で新しい値を入れる必要がありますが、実はテキストには書いてないのですが UPDATE だけでやることが可能です。次の UPDATE 文で id が 1 の人の remainder の値を 300 減らすことができます。

```
UPDATE user SET remainder=remainder-300 WHERE id=1
```

   pay2.php の最後で、支払先や金額の$_SESSION[] を消すようにしてください。

6. 前期のテキストの「5.15 送信前のチェック」を参考にして、pay.php に JavaScript の記述を追加して、「O.K.」ボタンをクリックすると支払金額が残額を超えていたら window.alert() を使って「残額が足りません。」と表示されるようにします。支払金額が残額以下ならば、これまで通り pay2.php に行くようにします。

9.4　QR 決済システム (4) 10/18

今回で QR 決済システムでの完成を目指します。これまでの課題ができていない人は、他の人の回答などを参考にしてまず追いつきましょう。

1. 支払いのテストは、ブラウザに支払いの URL を入力して行います。「http://mars..../index.php?to=2&pay=300」で支払先の利用者の ID は 2 で、支払金額は 300 円であることを示します。
2. index.php の最初で、URL に支払先があれば支払先を$_SESSION['to'] に入れます。支払先がなければ$_SESSION['to'] を削除してください。さらに URL に金額があれば、$_SESSION['pay'] に入れますが、無ければゼロを$_SESSION['pay'] に入れてください。次に$_SESSION['id'] があれば list.php へ行くようにします。list.php に行かない場合は、これまでのように、名前とパスワードの入力画面を出してください。
3. pay.php は、「支払わない」のリンクをクリックしたら index.php へ行くようにしてください。

4. list.php の最初に$_SESSION['to'] があったら pay.php へ行くところを追加してください。また次の三井さんの回答を参考に支払い明細の表が出るようにしてくだしてください。

```
<Table Border>
<Tr><Th>日時</Th><Th>相手</Th><Th>入金</Th><Th>出金</Th></Tr>
<?php
$sql="SELECT * FROM money WHERE uidf={$_SESSION['id']} OR
                                uidt={$_SESSION['id']}";
$result=$db->query($sql);
if (!$result) { die($sql."の実行ができません。"); }
foreach ($result as $data) {
   if ($data['uidf']==$_SESSION['id']){
      $sqq="SELECT * FROM user WHERE id=".$data['uidt'];
      $resull=$db->query($sqq);
      if (!$resull) { die($sqq."の実行ができません。"); }
      $datt=$resull->fetch(PDO::FETCH_ASSOC);
      echo "<Tr><Td>" ,date("Y/m/d H:i",$data['date']), "</Td><Td>",
           $datt['name'],"</Td><Td></Td><Td>".$data['amount']."</Td></Tr>\n";
   } else {
      $sqq="SELECT * FROM user WHERE id=".$data['uidf'];
      $resull=$db->query($sqq);
      if (!$resull) { die($sqq."の実行ができません。"); }
      $datt=$resull->fetch(PDO::FETCH_ASSOC);
      echo "<Tr><Td>" ,date("Y/m/d H:i",$data['date']), "</Td><Td>",
           $datt['name'],"</Td><Td>".$data['amount']."</Td><Td></Td></Tr>\n";
   }
}
$db->commit();
?>
</Table>
```

ただデータベースの検索の中に検索が入っているので、ここを function を利用して分かりやすくしてください。

```
<Table Border>
<Tr><Th>日時</Th><Th>相手</Th><Th>入金</Th><Th>出金</Th></Tr>
<?php
function getName($id) {
   global $db;

     中略

  echo $data['name'];
}

$sql="SELECT * FROM money WHERE uidf={$_SESSION['id']} OR

     中略

echo "<Tr><Td>" ,date("Y/m/d H:i",$data['date']), "</Td><Td>";
getName($data['uidt']);
echo  後略
```

のような感じに。日付を出す部分は if の前に持って来ればもう少し簡潔になると思います。

9.5　画像データベース 10/25

簡単な画像データベースを作成します。全て「Image」ディレクトリの中に作成してください。

1. 画像データを入れるデータベースファイル (data.db) を作成します。次のようなテーブル (photo) を作成します。

名前	データ型	備考
id	INTEGER	PK、AI
name	TEXT	
image	BLOB	

2. QR 決裁システムで使用した common.php をコピーします。ただし`$_SESSION` が出て来る部分は削除します。
3. 図 9.7 のような index.php を作成します。画像は最大幅が 200px になるように設定します。なお画像ファイルをドラッグ＆ドロップしても送信されないようにします。画像ファイルは PNG 形式のみアップロードすることにします。

図 9.7　作成例

4. index.php から送信された画像をデータベースに追加して、index.php に戻る upload.php を作成します。
5. 「`<Img src="display.php?id=1">`」のようにして呼び出すと photo テーブルの中の id=1 の画像を表示する display.php を作成します。
6. index.php の削除のリンクをクリックした際に画像データを削除する delete.php を作成します。削除したら index.php に戻るようにします。

9.6　YOLO による物体検出 11/1

今回の課題はアップロードした画像を YOLOv5 で物体検出させるものです。

1. 図 9.8 の左側のような yolo.htm を作成します。「認識」をクリックしたら yolo.php へ行くようにします。
2. yolo.php で図 9.8 の右側のようなものが表示されるようにします。

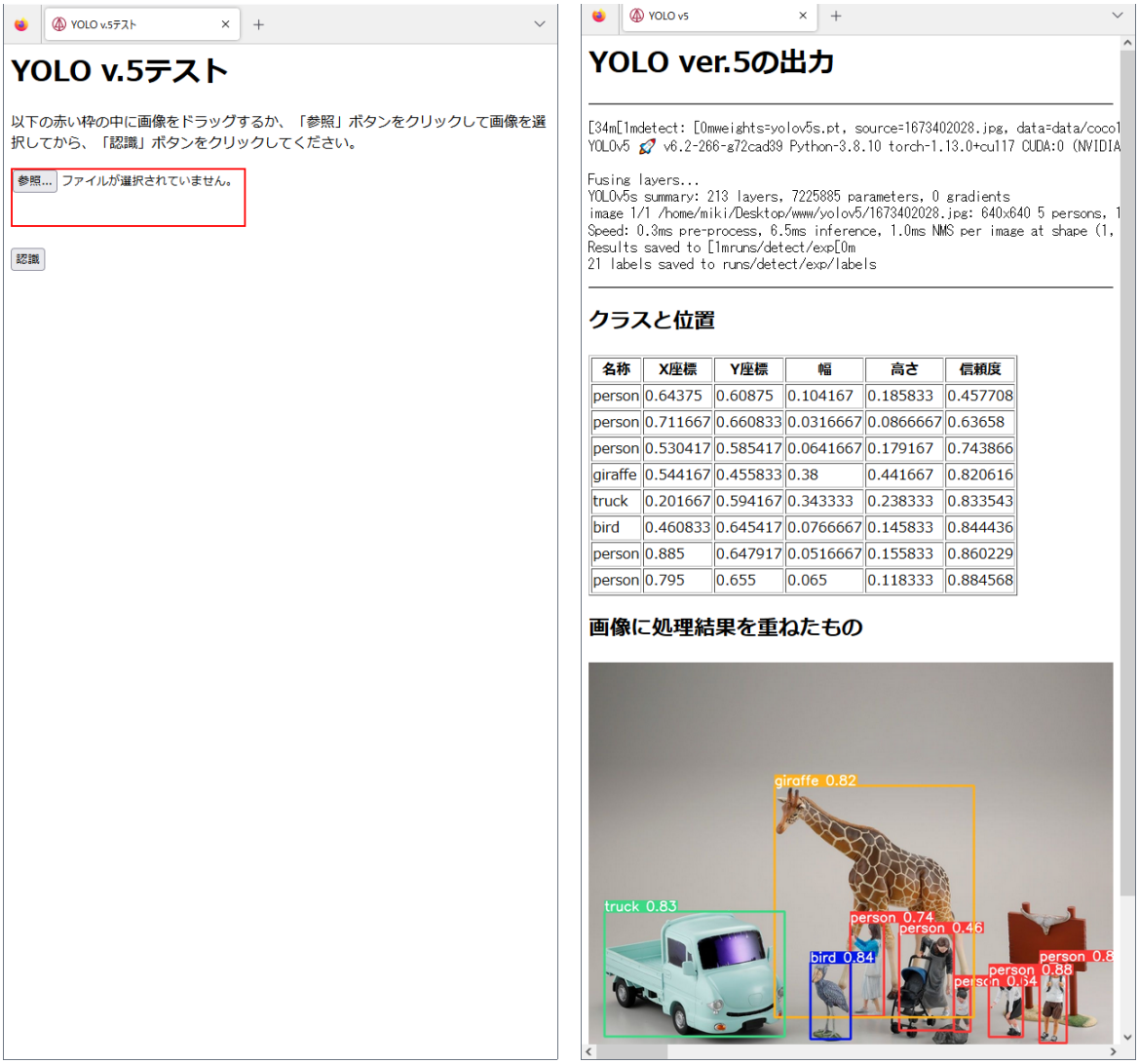


図 9.8 作成例

3. アップロードの際には「yolo5」の中にファイルを入れる必要があるので、

```
move_uploaded_file($_FILES['ufile']['tmp_name'],"yolov5/$fname$ext");
```

のようにアップロードしたファイルを保存する時に付ける名前の前に「yolov5/」を付けます。

4. 表のところで数値 (例えば 0) を名称 (例えば person) に直す必要がありますが、こちらから提供する class.php を取り込んで活用してください。
5. 写真画像は高解像度のものは大変大きく表示されることがあるので、画像の横幅がブラウザの幅より大きくならないようにしてください。

9.7　メールの定期送信 11/8

次の課題もメール関連でファイルが多数関連するので「Mail」と言うディレクトリを作成して、その中にこの課題のファイルも入れてください。

1. mars の自分のメールアドレスに一日に 3 回メールを送るものを作成します。ファイル名は aisatu.php にしてください。aisatu.php は実行する時間帯によって次のように異なる内容のメールを送るようにします。

時間帯	件名	本文
7:00 から 12:00 まで	おはよう！	今日も良い一日だとよいですね。
12:00 以降 17:00 まで	こんにちは！	明るいうちにがんばりましょう。
上記以外	こんばんは！	夜は早く寝ましょう。

2. cron を設定して aisatu.php を毎日 8:00、13:00、22:00 に呼び出すようにします。設定した内容を aisatu.php の`<?php` の次の行に、「//」を入力し、その後に入力しておいてください。なお、設定通りメールが届いたら、cron の設定は消しておきましょう。そうしないとどんどんメールが溜まってしまいます。

9.8　利用者登録の確認 11/15

利用者に自分の好きな名前とパスワードで使ってもらうシステムはあちこちで見られます。登録の際には、名前とパスワードの他に連絡用にメールアドレスの入力を求める事も多いと思います。その場合問題になるのは、メールアドレスが間違っていると連絡が取れないことです。対策の一つとして、登録の際に入力されたメールアドレスへメールを送り、それに対する応答を元に登録を行う事が考えられます。これによって登録された利用者のメールアドレスは、登録時には届いた事が確認できます。

1. 以下のファイルは「Mail」ディレクトリーの中に入れるようにします。
2. Image ディレクトリにある common.php と同じ内容の同じ名前のファイルを作成します。
3. 「DB Browser for Sqlite」を利用して data.db を作成します。テーブルは「user」の一つだけで、その内容は id INTEGER PK AI、name TEXT、password TEXT、address TEXT、status INTEGER とします。データを入れる必要はありません。
4. index.htm では図 9.9 のように名前、パスワード、メールアドレスを入力するところと、「登録」ボタンを設けます。「登録」ボタンをクリックしたら sendmail.php へ行くようにします。

図 9.9　作成例

5. sendmail.php ではまず data.db に入力内容を入れるようにします。名前→ name、パスワード→ password、メールアドレス→ address、0 → status。「DB Browser for Sqlite」を利用してちゃんと data.db に入ったかどうか確認しましょう。それから図 9.10 のように出るようにしましょう。

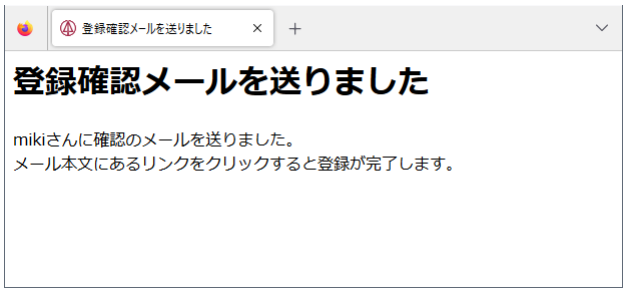


図 9.10 作成例

6. さらに sendmail.php で、前期のテキストの「6.5 INSERT 文」を参考に今データベースに入れたレコードの id の値を取り出して、echo で表示するようにしてみましょう。ちゃんと表示されたらこの部分は削除します。
7. そして sendmail.php でさらに入力されたメールアドレス先に、「登録を完了するために以下のリンクをクリックしてください。」と URL からなる本文のメールを送るようにします。URL は次に作成する touroku.php の URL の後に、6. で取り出した id の値を touroku.php で取り出せるような形にしたものを付けます。
8. touroku.php では、id の値をもとに、status の値を 1 に変更します。よってデータベースの中の status の値を見れば、確認済みの利用者かどうか分かるようになります。変更後に図 9.11 のように出るようにしてください。

図 9.11 作成例

9.9 円相場のグラフ 11/22

次の課題で環境センサーの測定値をデータベースに入れて、データベースの内容をグラフにして表示するものを作ります。今回はその後半の部分の練習になります。

1. 「IoT」と言うディレクトリを作成し、以下のファイルはその中に入れてください。
2. 「教材フォルダ」にある yen.db をコピーしてください。このデータベースファイルには souba と言うテーブルがあり、そこには id、date、yenHigh、yenLow と言う列があり、今年になってからの毎日の円相場の数字が入っています。
3. yengraph.php を作成して、yen.db の最初の 15 件のデータを元に図 9.12 のようなグラフが出るようにしてください。id が 1 以上かつ 15 以下で検索すると、最初の 15 件のデータが得られます。データベースの検索結果は日付と円がセットで順番に出て来るので、データベースから foreach を使って検索結果をまず変数に入れてグラフにすると良いでしょう。
4. 5 秒ごとに更新されるようにします。

5. 更新されるごとに、表示されるデータが後にずれるようにします。つまり、一番最初は 1/4 からですが、更新されると 1/5 からになり、さらに更新されると 1/6 からになるようにします。souba テーブルの id の値が 1 から順番になっていることを利用します。これを実現するには、更新後のページにどこから表示するのかを、URL を使って伝えましょう。そのままずっと放置すると yen.db に入っていないところまで行ってしまいますが、それに対応する必要はありません。
6. 「最初から」リンクの URL は「yengraph.php?id=1」になっています。これをクリックすると、グラフが 1/4 から表示されるようにしてください。

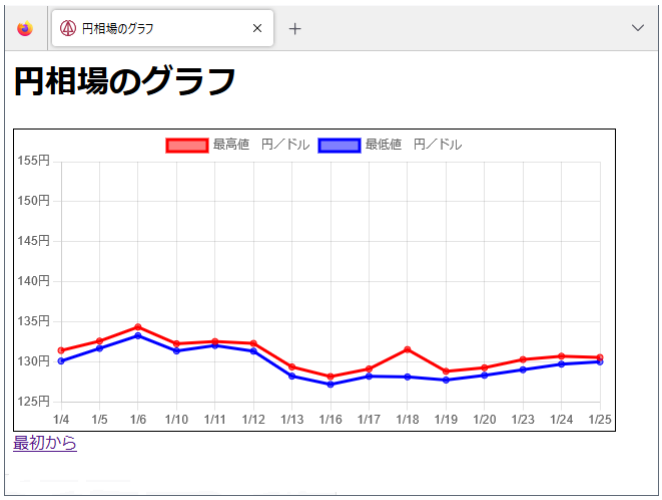


図 9.12　作成例

9.10　環境センサーのデータのグラフ化 11/29

環境センサーからのデータを受け取りグラフ化する課題です。今回の課題のファイルは「IoT」ディレクトリに入れてください。

1. データベース (data.db) を新規に作成します。テーブル名は「sensor」にして、フィールドとして「id」、「date」、「temp」、「humi」、「co2」、「bright」を追加します。データ型は「date」、「co2」、「bright」は「INTEGER」、「temp」と「humi」は「REAL」です。「id」のみ AU と PK を設定します。
2. データベースと接続するための common.php を作成します。
3. テキストの「4.5 環境センサーからの情報の取り込み」を参考にして、receive.php を作成します。何も echo する必要はありませんし、HTML のタグも不要です。`$_POST[ ]` で受け取ったデータを data.db に入れてください。
4. receive.php ができたら、`http://mars.mgt.sugiyama-u.ac.jp/Sensor/` で登録してください。
5. data.db のデータを元に図 9.13 のようなグラフを表示する co2graph.php を作成してください。一番新しいデータが最大 1 日分 (15 分ごとの平均) が表示されるようにしてください。また 60 秒ごとに自動更新されるようにします。両側に縦軸のあるグラフの作り方は「とほほの WWW 入門」の例を参考にしてください。なお時刻のところは、自動的に間引かれて表示されます。なお 15 分ごとの平均を求める SQL 文は、次のような感じになります。

```
SELECT date, AVG(temp) AS T, AVG(humi) AS H, AVG(co2) AS C FROM sensor
    WHERE date BETWEEN $first AND $last GROUP BY date/900
```

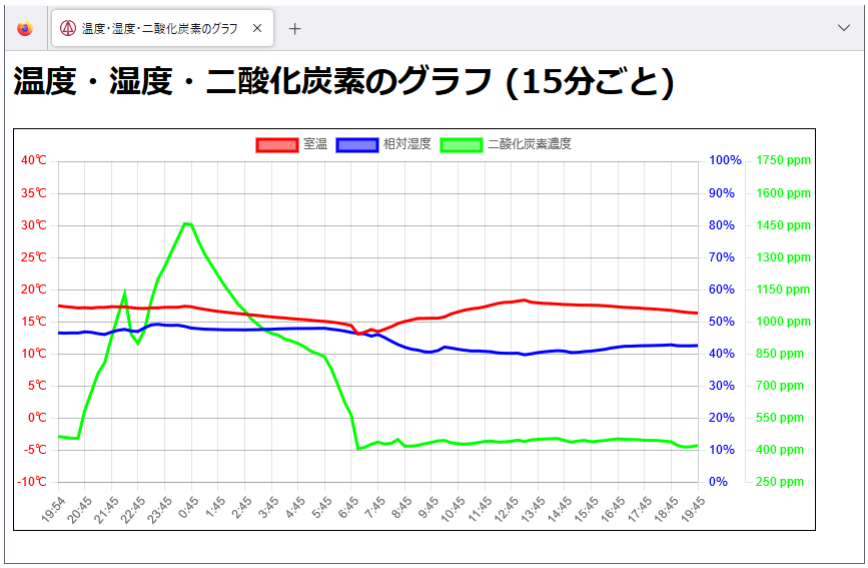


図 9.13　作成例

6. receive.php を修正してセンサーの二酸化炭素濃度が 1000 を超えたらメールを送って知らせるようにします。ただし一度 1000 を超えたら 800 未満になるまでメールは送りません。800 未満になったらメールで知らせます。本文には、co2graph.php の URL も入れてください。メールの送り先は mars でも自分のスマホでも構いません。データベースに alert と言うフィールドを追加して、1000 を超えたらずっと 1、800 未満になったらずっと 0 を入れるようにするとできます。

9.11　スクレイピングと POST でデータを送る 12/6

クレイピングを使って株価のデータを取り込むものと、POST 形式のデータを送る課題です。

1. 「株マップ.com」(`https://jp.kabumap.com/`) にアクセスして現在の日経平均株価が図 9.14 の左側ように表示されるような kabmap.php を作成せよ。
2. 「QR/user2.php」へ直接アクセスして名前が「miko」、パスワードが「99999」、残額が「10000」の利用者を登録する adduser.php を作成せよ。なお `file_get_contents()` で取ってきた内容はそのまま echo で出力すると図 9.14 の右側ようになる。

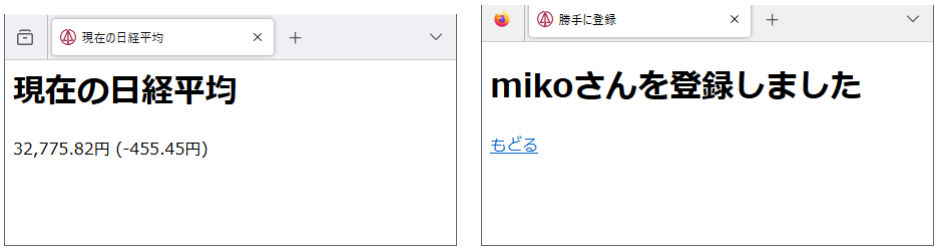
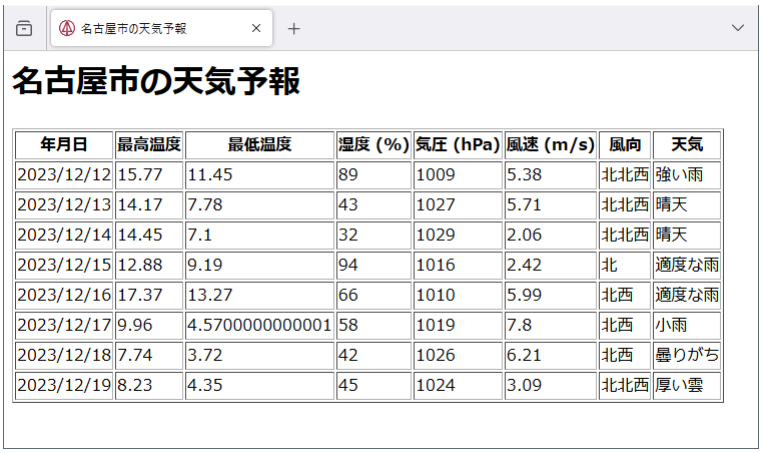


図 9.14　作成例

9.12　天気予報の表示 12/13

今回の課題は「Open Weather Map」から 1 週間の天気予報を受け取り、図 9.15 のような表の形で表示する weather.php を作成することです。



年月日	最高温度	最低温度	湿度 (%)	気圧 (hPa)	風速 (m/s)	風向	天気
2023/12/12	15.77	11.45	89	1009	5.38	北北西	強い雨
2023/12/13	14.17	7.78	43	1027	5.71	北北西	晴天
2023/12/14	14.45	7.1	32	1029	2.06	北北西	晴天
2023/12/15	12.88	9.19	94	1016	2.42	北	適度な雨
2023/12/16	17.37	13.27	66	1010	5.99	北西	適度な雨
2023/12/17	9.96	4.5700000000001	58	1019	7.8	北西	小雨
2023/12/18	7.74	3.72	42	1026	6.21	北西	曇りがち
2023/12/19	8.23	4.35	45	1024	3.09	北北西	厚い雲

図 9.15　作成例

9.13　シューティングゲームの作成 12/20

今回の課題は ChatGPT を利用して、JavaScript によるシューティングゲームを作成する事です。どのようなシューティングゲームするかはお任せします。作成したゲームは game.htm に、作成のための ChatGPT とのやり取りはコピペして game.txt に保存してください。なお、

- もし文字化けが見られる場合は、`<Meta Charset="UTF-8">`を`<Head>`〜`</Head>`の間に追加してください。
- 質問のときの token 数が 2,000 を超えるような場合は、gpt-3.5-turbo-16k か gpt-4 を使いましょう。(gpt-4 は 20 倍ぐらい高いけど)

9.14　志願理由書作成システム 1/10

今回の課題は ChatGPT を利用して志願理由書を作ってくれるシステムです。まず es.htm は図 9.16 のような画面です。

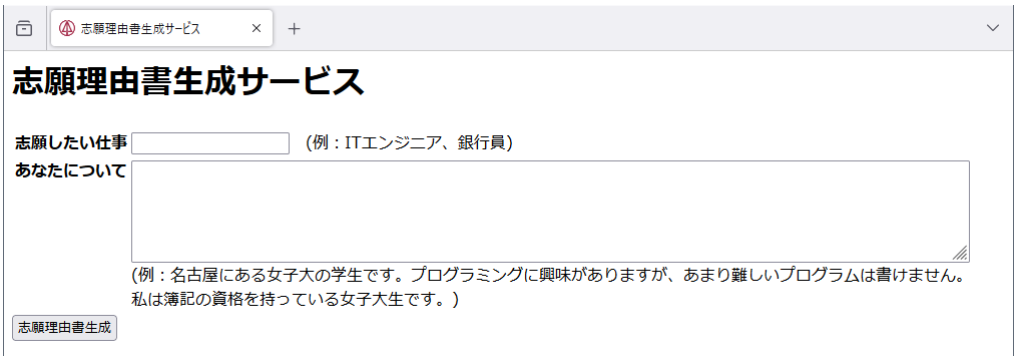


図 9.16　志望先など入力画面

ここで「志願したい仕事」と「あなたについて」の欄に入力して、「志願理由書生成」をクリックすると、es.php が呼び出されて、最初は図 9.17 のような画面になります。

そして、図 9.18 のような画面に変わって結果を表示します。注意点としては、

1. 図 9.17 の画面に出て来る画像は、自分で適当な動画 GIF の画像を探してきてください。違うデザインでも構いません。

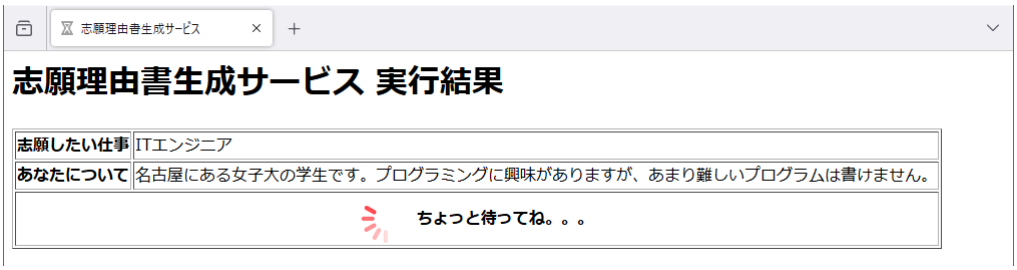


図 9.17　ChatGPT 応答待ち画面

2. 図 9.17 の画面の画像や「ちょっと待っててね。。。」は個別に消すのは面倒なので、`<Tr>`〜`</Tr>`ごと消すのが良いでしょう。
3. 図 9.17 の画面の表の左の列は何も指定がないと潰れるので、ちょっと HTML で対処してください。(Style よりも楽なので)
4. 図 9.18 の文字数は mb_strlen() で数えてください。
5. 図 9.18 の画面の文章は何も指定しないと 500 字を超えるような文になります。あまり長くならないようにしてください。(指定しても長い文が出て来ることがありますが。)
6. 図 9.18 の画面の「もどる」をクリックすると es.htm へ行くようにしてください。

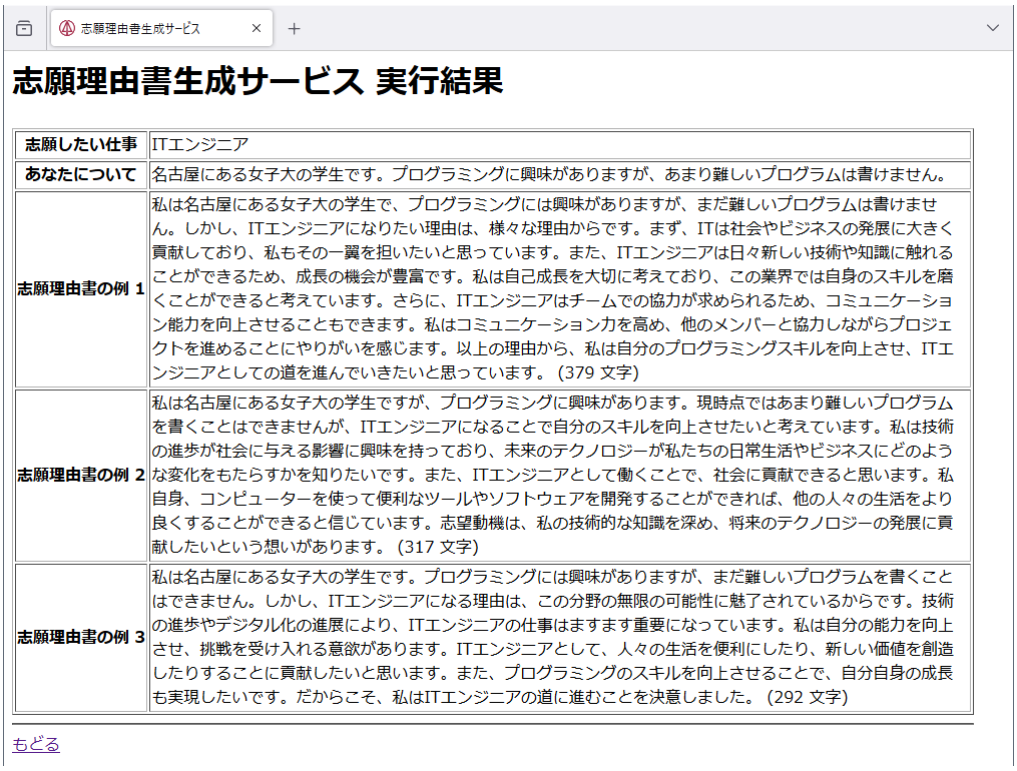


図 9.18　結果表示画面

9.15　API を利用した Chat システム 1/17

今回の課題は ChatGPT の API を利用して Chat をするシステムの作成です。まず chat.php は図 9.19 のような画面です。

1. SESSION 変数が使えるようにして、`$_SESSION['n']` が存在しないか、`$_GET['clear']` が存在する

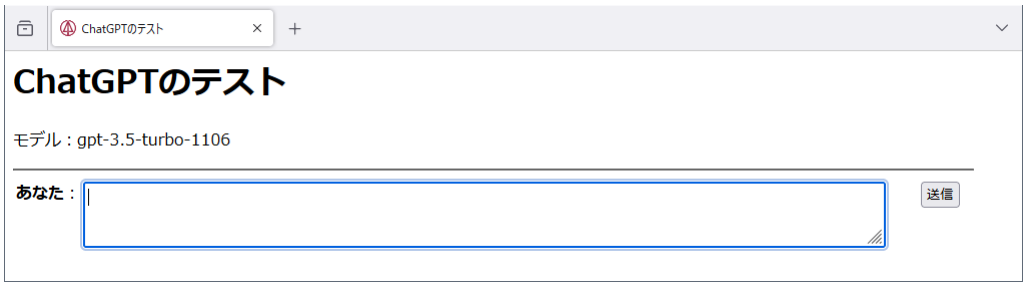


図 9.19　作成例

時に、$_SESSION['n'] に 1 を入れてください。

2. モデルは gpt-3.5-turbo-1106 を使うようにしてください。(実際に使用するのは chat2.php ですが)
3. 結果が画面幅いっぱいに広がると読みにくいので、<Body>の最大幅を 60em にしてください。
4. 入力欄の名前は「user」にしてください。
5. </Form>の後に次の JavaScript のコードを実行するようにしてください。

```
window.scroll(0,document.body.scrollHeight);
document.aaa.user.focus();
```

これによって画面が一番最後にスクロールされ、「user」と言う名前の入力欄が選択されます。

6. 「送信」ボタンをクリックすると、「送信」ボタンを非表示にし、代わりに動画 GIF が表示されるようにした上で、chat2.php へ送信してください。
7. $_SESSION['n'] がゼロより大きい時は、図 9.20 のように「Chat のやり取りを初期化する」ボタンが表示されるようにしてください。

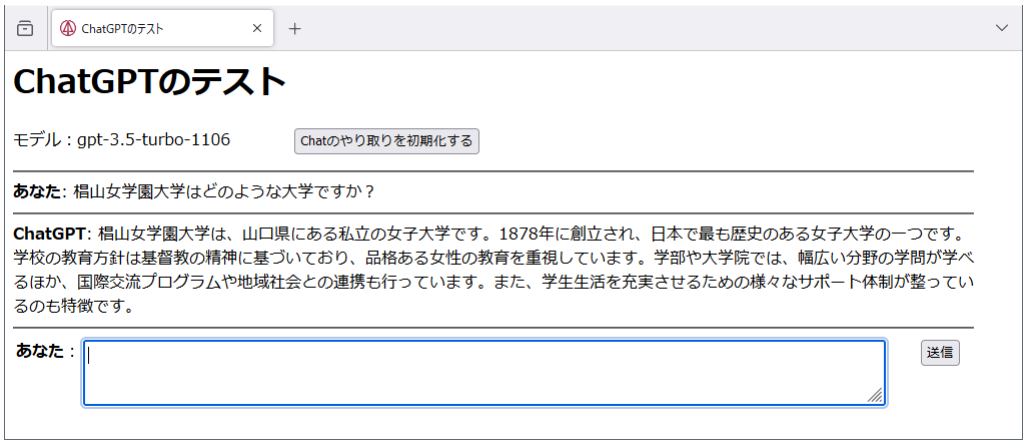


図 9.20　作成例

8. 「Chat のやり取りを初期化する」ボタンをクリックしたら、JavaScript で以下を実行するようにしてください。

```
location.href="chat.php?clear=yes";
```

9. ボタンの下のこれまでのやり取りの、「あなた」の分の内容は$_SESSION['chatmsg'][0]～に、「ChatGPT」の分の内容は$_SESSION['botmsg'][0]～に入っているものとします。それぞれ$_SESSION['n'] に入っている数だけあるものとします。

次に、chat2.php は PHP のみです。

1. $_POST['user'] が存在しないか、$_POST['user'] が空か、$_SESSION['n'] が存在しない場合は、chat.php へ行くようにします。
2. これまでのやりとりと最後に入力内容を付け加えたものを$role に入れるために次のようにします。この変数をテキスト p.45 のリスト 6 の 8 行目の「=>」の次の [...] の代わりに指定します。

```
$role=array();
for ($i=0;$i<$_SESSION['n'];$i++) {
   $role[]=array('role' => 'user', 'content' => $_SESSION['chatmsg'][$i]);
   $role[]=array('role' => 'assistant', 'content' => $_SESSION['botmsg'][$i]);
}
$_SESSION['chatmsg'][$_SESSION['n']]=$_POST['user'];
$role[]=array('role' => 'user', 'content' => $_POST['user']);
```

3. 上記を参考に$_SESSION['botmsg'] に返ってきた内容を入れてから、$_SESSION['n'] の内容を 1 増やします。
4. 最後に chat.php へ行くようにします。

索引

A

at: 実行予約をする (Unix) 16

C

cron: 定期的実行予約をする (Unix) 16

E

explode(): 文字列を区切り文字をもとに分解する (PHP) 13

M

mb_send_mail(): メールを送る (PHP) 14

P

pclose(): Unix のコマンドとのやり取りを終える (PHP) 11
popen(): Unix のコマンドを実行する (PHP) . . 11

S

system(): Unix のコマンドを実行する (PHP) . 11

U

Unix のコマンドを実行する (PHP) 11

W

wget: web ページを取り込む (Unix) 15

Y

YOLO: 物体検出システム 11

さ

実行予約をする (Unix) 16

た

定期的実行予約をする 16

は

物体検出システム 11

ま

メールを送る (HTML) 14
メールを送る (PHP) 14
文字列を区切り文字をもとに分解する (PHP) . . 13