基礎プログラミングII

第7回　CGIと情報システム (1)

メディア情報コース
平居　悠（ひらい　ゆたか）

到達目標

プログラミングを用いた実践的なデータ処理と情報システムの構築

- 定式化された処理を**関数の形**で記述し利用することができるようになる
- 再帰などの**アルゴリズム**を理解し問題に適用できるようになる
- 基礎的な**CGI**の仕組みの理解を通して**Webインターフェース**を設計できるようになる
- 多様な**社会事象への適用**を設計できるようになる
- 現実社会の課題に対応する**情報システム**を設計・作成できるようになる
- **生成AI**を学習の道具として利用できるようになる

前回

第１回　生成AIを効果的に利用した学習法
第２回　メソッド定義と効果的活用（関数的処理）
第３回　メソッド定義と効果的活用（データ集合処理）
第４回　メソッド定義と効果的活用（再帰的アルゴリズム）
第５回　変数のスコープ・クラス設計
第６回　専門演習紹介
第７回　CGIと情報システム（1）
第８回　CGIと情報システム（2）
第９回　チーム課題準備期間（1）
第１０回　チーム課題準備期間（2）・trr試験
第１１回　予選発表
第１２回　代表発表
第１３回　合同成果発表会
第１４回　期末試験

前回の目標

専門演習ではどのようなことが行われているのかを知る。

専門演習について

- ３年次から始まる卒業論文執筆に向けたゼミ（必修）。
- 公益大の専任教員を1名選択し、その研究室に所属して2年間研究を行う。

専門演習選択ガイダンス

日時：11月20日 (木) 10:10~10:30
場所：301教室

専門演習公開授業

日程：11月25日 (火)～12月8日 (月)
日時・場所：ガイダンス資料を参照のこと

自分に合った専門演習を選ぶには

- ガイダンスで配布されるシラバスをよく確認する。
- 教員紹介のページ (https://www.koeki-u.ac.jp/about_us/kyouin/teaching_staff.html)で興味のある教員の研究内容、論文を調べる。荘内日報「公益の風」も読むと良い。
- 候補となる先生方に連絡を取り、面談をしてもらう。

平居研究室(H-2研究室)の場合

- 2~4限は在室のことが多い
- 居室のドアが空いていれば面談対応可
- 月曜3限、金曜3, 4限は授業
- 12月2日(火)は終日不在
- メール (yutaka.hirai@koeki-u.ac.jp)で予約を取った方が確実

タイピング練習スケジュール

第２回　　ホームポジション
第３回　　ローマ字
第４回　　英語初級
第５回　　日本国憲法
第６回　　ホームポジション
第７回　　ローマ字
第８回　　英語初級
第９回　　日本国憲法
第１０回　日本国憲法（trr試験、合格点：200点）

タイピングの練習 (ローマ字)

1. ブラウザを起動し、https://www.koeki-prj.org/trr/ に繋ぐ。
2. 学籍番号（Cは大文字、省略なし８桁）を入力する。
3. Koeki MAILに届いたパスコードをPasscode: 欄に入力する。

今回

第１回　生成AIを効果的に利用した学習法
第２回　メソッド定義と効果的活用（関数的処理）
第３回　メソッド定義と効果的活用（データ集合処理）
第４回　メソッド定義と効果的活用（再帰的アルゴリズム）
第５回　変数のスコープ・クラス設計
第６回　専門演習紹介
第７回　CGIと情報システム（1）
第８回　CGIと情報システム（2）
第９回　チーム課題準備期間（1）
第１０回　チーム課題準備期間（2）・trr試験
第１１回　予選発表
第１２回　代表発表
第１３回　合同成果発表会
第１４回　期末試験

今回の目標

CGIの仕組みについて理解し、簡単なプログラムを書けるようになる。

今回学ぶこと

1. CGI入門
2. 検索主体のCGIプログラムの作成
3. 紙芝居のようにページを進めるCGI

今回の問い

- CGIとは何か？
- データベース検索をCGI化するにはどのようにすれば良いか？
- １つのCGIプログラムで紙芝居のように順次移り変わるWebページを生成するにはどうすれば良いのか？

今回学ぶこと

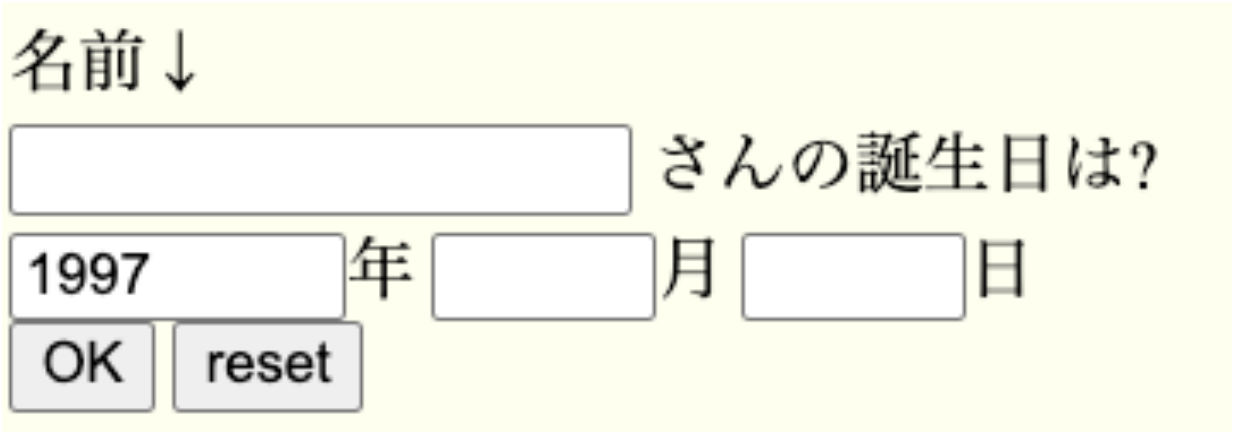
1. CGI入門
2. 検索主体のCGIプログラムの作成
3. 紙芝居のようにページを進めるCGI

CGI (Common Gateway Interface)とは

HTML文書内にあるデータ入力窓とそれらを受け渡すスクリプト名を書いておき、ボタンを押すとそのスクリプトに入力値が渡るような仕組み

CGIの仕組み

CGIのデータ入力窓の集合を**入力フォーム**という。



広瀬先生のページ「CGI入門」「CGIのしくみ」の下にある入力フォームに値を入力もしくは選択して「OK」ボタンを押してみよう。

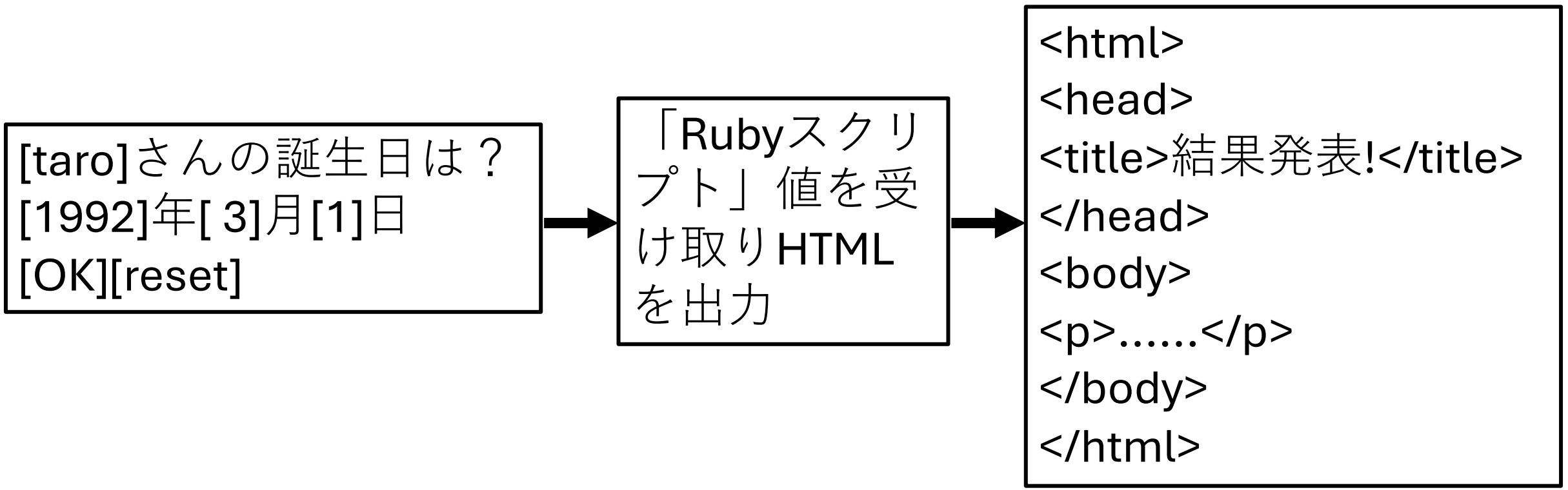
CGIの仕組み

前ページの入力窓に入れられた値は全て**入力名**という特別な変数CGIスクリプトに送られる。

CGIの仕組み

自分で作るプログラムではその変数を
受け取りそれに応じた処理をした上で、
結果を表すHTML文書を出力する。

CGIの仕組み


[taro]さんの誕生日は？
[1992]年[3]月[1]日
[OK][reset]
「Rubyスクリプト」値を受け取りHTMLを出力
<html>
<head>
<title>結果発表!</title>
</head>
<body>
……
</body>
</html>

CGIの作り方

CGIでは、利用者にデータを入力してもらうための**HTML文書**とそこから送られて来るデータを受け取って処理をする**プログラムファイル**を作る。

CGIの作り方

Webサーバに対して「ここでCGIプログラムを利用する」ことの宣言を行う必要がある。

CGIの作り方

1. CGI利用宣言→.htaccessファイルの作成
2. 入力フォームのHTML文書作成
3. 処理プログラムの作成

CGIプログラムの利用宣言

「この場所でCGIを利用する」ためのディレクトリを作成し、そこに.htaccessという名前のファイルを作成する。

CGIプログラムの利用宣言

ここでは、~/public_html/mycgiというディレクトリを作成する。

```
cd ~/public_html
mkdir mycgi
```

CGIプログラムの利用宣言

次にEmacsで~/public_html/mycgi/.htaccess ファイルを新規に開き、以下の内容を記述して保存する。

```
AddHandler cgi-script .rb
Options +ExecCGI
AddType “text/html; charset=utf-8” .html
```

CGIプログラムの利用宣言

```
AddHandler cgi-script .rb
Options +ExecCGI
AddType “text/html; charset=utf-8” .html
```

1, 2行目：このディレクトリに作成する.rbという名前で終わるファイルはCGIスクリプトと認識される。

CGIプログラムの利用宣言

```
AddHandler cgi-script       .rb
Options +ExecCGI
AddType “text/html; charset=utf-8”.html
```

3行目：HTMLファイルの文字コードがUTF-8であることを宣言する。

HTML文書の構成

入力フォームを構成するHTML文書を作る。入力フォームはform要素を使って記述する。

```
<form method="メソッド" action="./スクリプト">
~~~
</form>
```

メソッドの部分はGETまたはPOSTのどちらかを指定する。文章など、長いデータを入力させたいときはPOSTを指定する。

HTML文書の構成

次ページのform要素例は

- namaeという入力名で最大40字の文字列を
- bloodという入力名でA, AB, B, Oのいずれかの選択肢を

入力させ、./btype.rbというスクリプトに入力値を渡すためのHTML記述である。

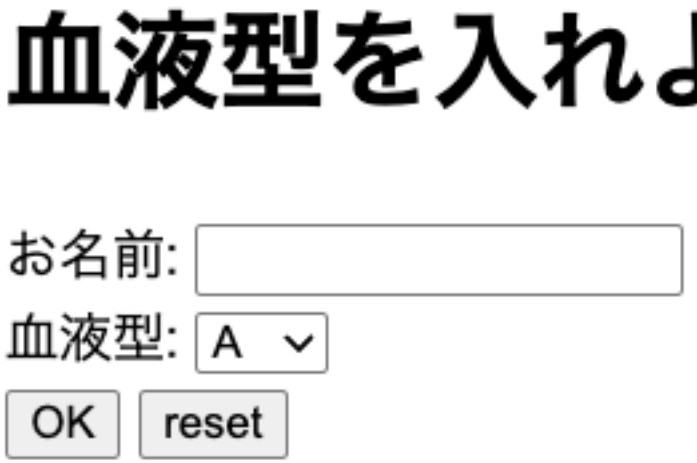
HTML文書の構成

```
<form method=“POST” action=“./btype.rb”>
<p>お名前：<input name=“namae” type=“text” maxlength=“40”><br>
血液型：<select name=“blood”>
    <option>A</option>
    <option>AB</option>
    <option>B</option>
    <option>O</option>
<input type=“submit” value=“OK”>
<input type=“reset” value=“reset”><br></p>
</form>
```

HTML文書の構成

前ページ入力フォームのみを
含むHTML文書の例：

http://roy.e.koeki-u.ac.jp/~yuuji/2025/pf2/cgi/btype.html



ソース：http://roy.e.koeki-u.ac.jp/~yuuji/2025/pf2/cgi/src/btype.html

CGIスクリプトの構成

CGIスクリプトとなるRubyプログラムでは、最低限以下の内容を記述して作成する。

```
#!/usr/bin/env ruby
# coding: utf-8


require ‘cgi’
c = CGI.new(:accept_charset => “UTF-8”)
print “Content-type: text/html; charset=UTF-8\n\n”
```

CGIスクリプトの構成

CGIスクリプトとなるRubyプログラムでは、最低限以下の内容を記述して作成する。

```
#!/usr/bin/env ruby
# coding: utf-8


require ‘cgi’
c = CGI.new(:accept_charset => “UTF-8”)
print “Content-type: text/html; charset=UTF-8\n\n”
```

変数cはどんな名前でも良い。このあとにHTML文書となり得る文字列を順次出力していく。

CGIスクリプトの構成

requireは、プログラムに必要な外部モジュールを現在のプログラムに取り込む記法で、

require ‘cgi’

はCGIプログラムを組むのに便利なクラスが定義された**cgi.rb**を取り込む宣言である。これでCGIクラスが使えるようになる。

CGIスクリプトの構成

CGIプログラムに渡された入力値を取り出すには、CGI.newを受け取った変数のハッシュ値として取り出す。

Form入力されたパラメータnamaeとbloodをHTML文書形式で出力するRubyプログラム

```
#! /usr/bin/env ruby
# coding: utf-8

require ‘cgi’
c = CGI.new(:accept_charset => “UTF-8”)
print “Content-type: text/html; charset=UTF-8\n\n”

name      = c[“name”] #フォームに記入されたnameの値を取り出す
bt        = c[“blood”] #フォームに記入されたbloodの値を取り出す

print << EOF
<!DOCTYPE html>
<html>
<head><title>Blood type</title></head>
<body>

<h1>#{name}さんの血液型</h1>

<p>#{name}さんは#{bt.upcase}型ですね！</p>
</body>
</html>
EOF
```

ヒアドキュメント(<<)とは

<<の次の行から、<<の直後に書かれた単語が見つかる行までをまとめて１つの文字列として括る記号のこと。

ヒアドキュメント

改行文字を含む長文文字列を扱いたいときに便利な記法。文字列中に“”や‘’を自由に入れることもできるため、“”を多様することの多いHTML文の出力に有用

#{...}

その部分をRubyの式としてその値に置き換える働きを持つ。

#{…}

例えば、

```
print “1+2は#{1+2}です\n”
```

とすると、

```
1+2は3です
```

と出力される。HTML出力文に変数結果を入れたい場合に有用。

処理の切り替え

入力した値による出力を切り替える例を作成してみよう。

入力されうる値をキーとして、対応する文字列をハッシュで登録しておき、メッセージ出力部分をハッシュの値を引くことによって切り替える。

```
#!/usr/bin/env ruby
# coding: utf-8

require 'cgi'
c = CGI.new(:accept_charset => "UTF-8")
print "Content-type: text/html; charset=UTF-8\n\n"

name = c["namae"]
bt   = c["blood"]

kekka = {# A, B, AB, O に対応する値をHashで設定する
 "A"        => "えーっすなあ",
 "AB"       => "だぶる!",
 "B"        => "びびっとね",
 "O"        => "おーすげー",
}
kekka.default = "人間なの??” # Hashのデフォルト値(キーがない場合の値)

bt.upcase!          # 念のため大文字に変えておく
```

```
# ここ↓の <<EOF の部分が次の行からEOF行手前までの文字列に置き換わる
printf(<<EOF, bt, kekka[bt])  # 8行下の%sに入る
<!DOCTYPE html>
<html>
<head><title>Blood type</title></head>
<body>

<h1>#{name}さんの判定結果</h1>

<p>%s型のあなたは%s</p>
</body>
</html>
EOF
```

CGIプログラムのデバッグ手順

1. ターミナルで直接動かしてエラーをなくす
2. ローカルPC（目の前で使っているPC)で動かして詳細デバッグ
3. 本番サーバで動かして最終調整

(1) ターミナルでテスト

btype.rbでnameに「太郎」を、bloodに「A」を代入した状態でデバッグする場合は以下のように起動する。

```
echo 'name=太郎&blood=A' | ./btype.rb
```

(1) ターミナルでテスト

つまり、入力名の値を

‘入力名$_1$=値$_1$&入力名$_2$=値$_2$&入力名$_3$=値$_3$&…’

の形式でechoコマンドで出力し、実行プログラムに送り込めばよい。

(1) ターミナルでテスト

まとめると、以下のようなプログラム起動で確認する。

echo '入力名$_1$=値$_1$&入力名$_2$=値$_2$&入力名$_3$=値$_3$' | ./作成中のプログラム.rb

正しいと思われるHTML文が出力されたら
次の手順に進む。

(2) ローカルWebサーバでのテスト

Rubyには簡易Webサーバ機能がある。
https://www.koeki-prj.org/~yuuji/2025/pf2/cgi/src/webserv.rb
設置方法：

1. 上記ファイルをテストしたいCGIプログラムと同じディレクトリにこのファイルを保存し実行属性をつける：chmod +x webserv.rb
2. 起動する: ./webserv.rb
3. http://localhost:XXXXのXXXXの部分に注意してブラウザで繋ぐ：http://localhost:指定された番号/自分が作ったもの.rb

(2) ローカルWebサーバでのテスト

保存したwebserv.rbを起動すると以下のようなメッセージが出る：

chmod +x webserv.rb

./webserv.rb

http://localhost:*1234/ファイル名*
で接続してください

(2) ローカルWebサーバでのテスト

もしエラーが出ると、webserv.rbを起動したターミナルにエラーメッセージが出るので自分の作ったプログラムに出された部分を選んで確認し、修正する。

(3) 正式WebサーバのURLで閲覧してテスト

前ページのような起動方法で、正しくHTML文書が出力されたら正式WebサーバのURLで開く。

URLの例：

http://roy.e.koeki-u.ac.jp/~cユーザ名/mycgi/ファイル名

(3) 正式WebサーバのURLで閲覧してテスト

ブラウザ経由でアクセスしてエラーになる場合は、Webサーバ(roy)にログインし/usr/local/apache2/logs/error_logファイルの最後の方を**tail –f**で参照する。

ssh roy　　←ログインパスワードを入れる

tail –f /usr/local/appache2/logs/error_log

デバッグが終わったらtailをC-cで止め、royからログアウトする。

フォームで使える入力用要素

input要素

なんらかの値を入力させる場合に、**input要素**に入力方法と入力名を指定する。

入力方法は**type属性**で、入力名は**name属性**で指定する。

input要素：text

1行だけの文字列入力

```
<p><label>メッセージをどうぞ：
<input name="v_text" type="text" value="Hello" size="40">
</label></p>
```

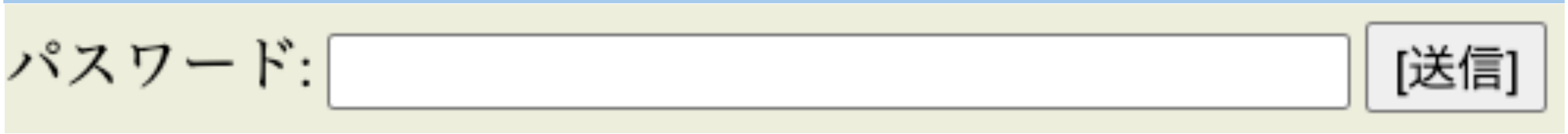
メッセージをどうぞ: Hello [送信]

入力名v_textに初期値“Hello”を入れて40字分の枠を作る。初期値を省略すると空欄になる。

input要素：passwd

textと同様だが、入力文字をアスタリスクで隠す。

```
<p><label>パスワード：
<input name=“v_passwd” type=“passwd” size=“20” maxlength=“12”>
</label></p>
```



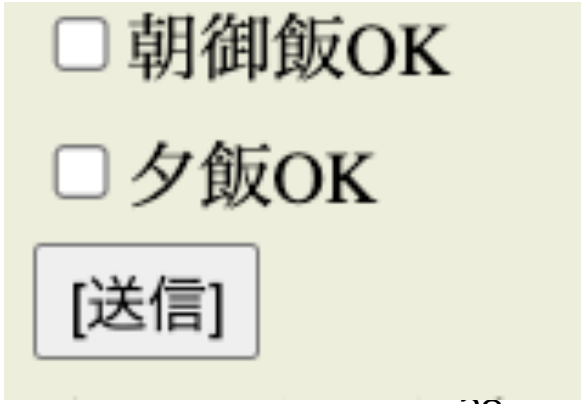
入力名v_passwdに最大文字数12字のパスワードをいれる。ただし、パスワードの値がCGIプログラムに渡されるときは**暗号化されない。**

input要素：checkbox

ONのときの値とOFFのときの値をvalue属性で指定する。

```
<p><label><input name="v_checkbox_1" type="checkbox" value="morning">朝御飯OK</label><br></p>
<p> <input name="v_checkbox_2" type="checkbox" value="evening">夕飯OK </p>
```

「朝御飯OK」がチェックされていたら入力名v_checkbox_1を“morning”に、「夕飯OK」がチェックされていたら入力名v_checkbox_2に“evening”を代入させる。



input要素：checkbox

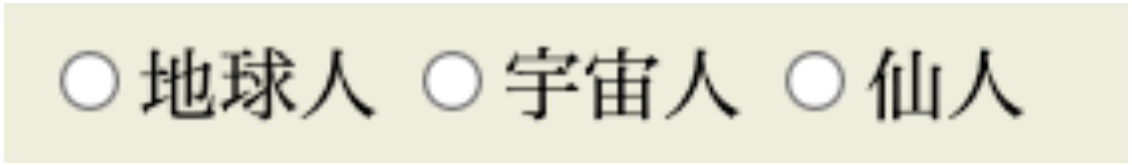
```
<p><label><input name="v_checkbox_1" type="checkbox" value="morning">朝御飯OK</label><br></p>
<p> <input name="v_checkbox_2" type="checkbox" value="evening">夕飯OK </p>
```

- 「朝御飯OK」はlabel要素で結びついているので、「朝御飯OK」をクリックすればチェックが入る。
- 「夕飯OK」はボタンをボタンをクリックしないとチェックが入らない。

input要素：radio

ラジオボタンを作成する。name属性で同じ入力名を持つもの同士がグループ化され、どれか一つをチェックすると他はクリアされる。

```
<p>
<label><input name="v_radio" type="radio" value="1">地球人</label>
<label><input name="v_radio" type="radio" value="2">宇宙人</label>
<label><input name="v_radio" type="radio" value="3">仙人</label>
</p>
```

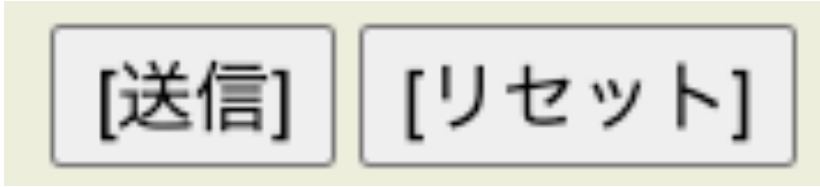


例えば「仙人」を選ぶと入力名v_radioに"3"が代入される。どれも選択されていない場合は空文字列が代入される。

input要素：submit, reset

入力値の送信またはリセットボタンを作成する。

```
<p>
<label><input type=“submit” value=“[送信]”>
<label><input type=“reset” value=“[リセット]”>
</p>
```



input要素：hidden

隠れ変数を設定する。CGIプログラムに必ず渡したい値を見えないところで指定する。

```
<input name="v_hidden" type="hidden" value="第3ステージ">
```

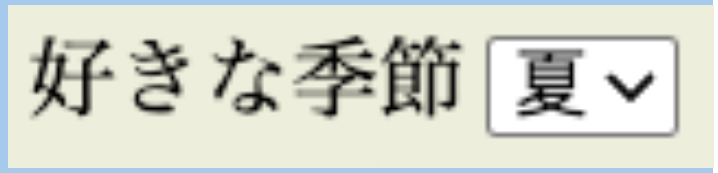
上記を書いてもWebページになにも表示されないがCGIプログラムには入力名v_hiddenに「第3ステージ」という値が代入される。

select要素

選択肢を提示してそのうち1つまたは複数を選ばせる。

好きな季節

```
<select name="v_select">
  <option>春</option>
  <option value="Summer" selected>夏</option>
  <option value="Autumn">秋</option>
  <option value="Winter">冬</option>
</select>
```

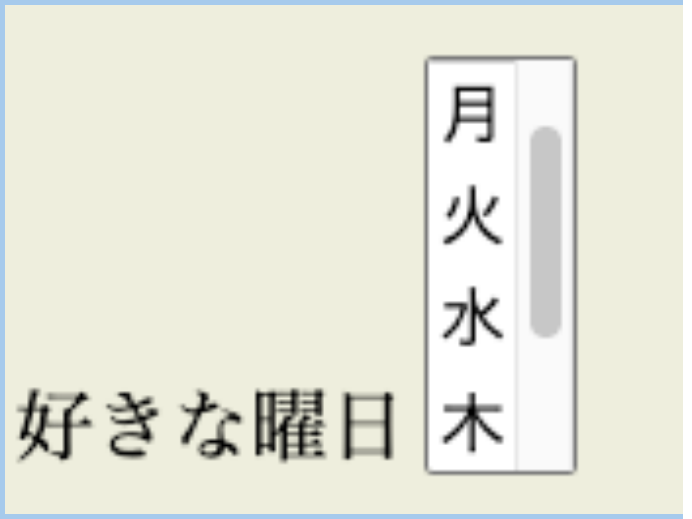


- value指定のない「春」を選択すると「春」そのものが値となる。
- selectedを付けておいたオプションが選択初期値となる。
- selectedを付けなければ先頭項目が選択された状態で始まる。

select要素

multiple属性を付けると複数選択が可能になる。Firefoxで複数選択するにはCtrlクリックで行う。

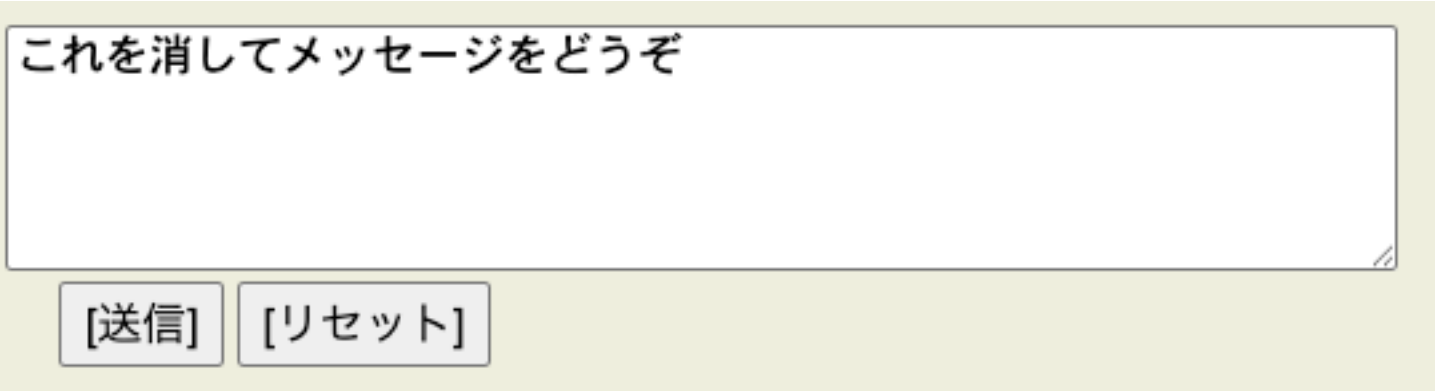
```
好きな曜日
<select multiple name="v_select_multi">
 <option value="Sunday" selected>日</option>
 <option value="Monday">月</option>
 <option value="Tuesday">火</option>
 <option value="Wednesday">水</option>
 <option value="Thursday">木</option>
 <option value="Friday">金</option>
 <option value="Saturday" selected>土</option>
</select>
```



テキストエリア

長文を入力させるためにはtextareaを用いる。入力エリアの行と桁数はそれぞれrows、cols属性で指定する。

```
<textarea name="v_textarea" cols="60" rows="5">
これを消してメッセージをどうぞ
</textarea>
```



その他のフォーム

フォーム制御の詳細はHTML Standard 4.10フォーム(https://momdo.github.io/html/forms.html#forms)が参考になる。

授業内課題

input要素に名前を入力、select要素から好きな季節を選択してOKボタンを押すと選んだ季節に応じて異なるメッセージを出力するRubyCGIプログラムseason.htmlとseason.rbを~/public_html/mycgi/内に作成し動くURLをリンクせよ。

s4基礎プロII(F)の「#07 pf2水 授業内課題 (season)」に指示どおり書き込む。提出後修正して良い。その場合は「編集」リンクから書き換えること。

今回の問い

- CGIとは何か？
  - HTML文書内にあるデータ入力窓とそれらを受け渡すスクリプト名を書いておき、ボタンを押すとそのスクリプトに入力値が渡る仕組み
- データベース検索をCGI化するにはどのようにすれば良いか？
- １つのCGIプログラムで紙芝居のように順次移り変わるWebページを生成するにはどうすれば良いのか？

今回学ぶこと

1. CGI入門
2. **検索主体のCGIプログラムの作成**
3. 紙芝居のようにページを進めるCGI

コンソールプログラム

コマンドラインで起動し、入出力を
すべてコマンドラインで完了させる
プログラム

コンソールプログラム

コンソールプログラムで値を利用者から読み取るときは主にgetsを利用しその結果を変数に代入する。

```
STDERR.print(“年齢を入力してください：”)
age = gets.to_i
```

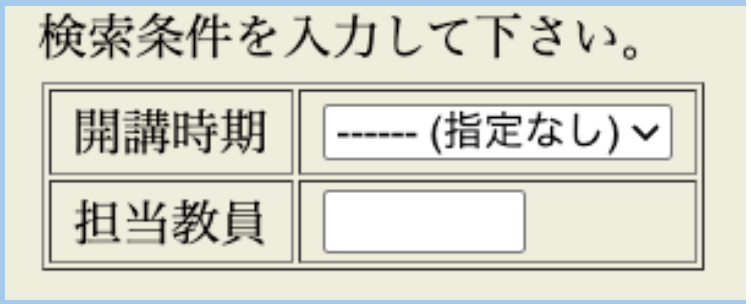
コンソールプログラムのCGI化

前ページのようなものをCGI化するには、入力に関わる部分をCGIパラメータから受け取るように変え、出力する部分をHTML形式で書き出すように変更する。

コンソールプログラムのCGI化

履修科目データベース検索をCGI化する。「開講時期」をjiki、
「担当教員」をkyoiで受け渡すHTMLフォームは以下の通り。

```
<form method="POST" action="db-cgi.rb">
 <p>検索条件を入力して下さい。</p>
 <table border="1">
  <tr><td>開講時期</td><td><select name="jiki">
                                  <option value="">------ (指定なし)</option>
                                  <option>春学期</option>
                                  <option>秋学期</option>
                                 </select></td></tr>
  <tr><td>担当教員</td><td><input name="kyoi" size="8"></td></tr>
 </table>
</form>
```



コンソールプログラムのCGI化

入力される値をjiki, kyoiから受け取るRubyプログラムは以下の通り。

```
require 'cgi'
c = CGI.new(:accept_charset => "UTF-8")
term = c["jiki"]
whom = c["kyoi"]
```

この結果出力をHTML形式で行えばCGI化は完了する。

search-cgi.rb 前半 （入力部分）

```
#!/usr/bin/env ruby
# coding: utf-8
Encoding.default_external = 'utf-8'     # UTF-8のCSVファイルを読むため
require 'cgi'
require 'csv'
c = CGI.new(:accept_charset => "UTF-8")

#データベース読み取り処理は全く同じ
csv = CSV.read("syllabus.csv", headers: true)
db  = csv               # あとで全データを使う場合にそなえcsv変数は温存

# 検索パターンはHTMLフォームへの入力値を取得する
term = c["jiki"]
whom = c["kyoi"]

if term > ""                                    # 開講時期指定に何か文字列を入れたなら
 ptn = Regexp.new(term)     # 文字列を正規表現に変換
 db = db.select {|row| ptn =~ row["開講時期"]}
end
if whom > ""                                    # 担当教員指定に何か文字列を入れたなら
 ptn = Regexp.new(whom)   # 文字列を正規表現に変換
 db = db.select {|row| ptn =~ row["担当教員"]}
end
```

search-cgi.rb後半（出力部分）

Rubyプログラムの“”内でHTMLの“”を使う際は、\“\”とする。

```
print("Content-type: text/html; charset=utf-8

<!DOCTYPE html>
<html>
<head><title>検索結果</title>
<style type=\"text/css\">
<!--
 tr.head {background: cyan;}
 th {width: 8em;}
 -->
</style>
</head>
<body>")
puts("<h1>該当科目一覧</h1>")
puts("<table border=\"1\">")
db.each {|row|
  printf("<tr class=\"head\"><td colspan=\"2\">%s</td></tr>\n",
       row["科目名"])
  row.each {|key, value|
    printf("<tr><th>%s</th><td>%s</td></tr>\n", key, value)
  }
}
puts("</table>")
puts("</body>
</html>")
```

入力フォームの自動出力

検索オプションが多数になるとoption要素一覧を用意するのが困難になる。

データの値に関連する選択肢を入力フォームにいれる場合、入力フォーム自体もデータの値を管理するCGIプログラムで生成すると良い。

search-cgi.rbで読み込むデータベース

値

ハッシュ

科目名	基礎プログラミング
科目コード	154
担当教員	鳥海三輔
開講時期	春学期
概要	プログラミングを通じて問題解決能力の向上を図る。

ハッシュ

科目名	応用プログラミング
科目コード	254
担当教員	飯森大和
開講時期	秋学期
概要	C言語を通じて問題解決に最適なアルゴリズムを適用する方策を学ぶ。

ハッシュ

科目名	応用もっけ論
科目コード	39
担当教員	酒田育造
開講時期	春学期
概要	その地域の「もっけ」を理解し、深い交流のきっかけとする。

search-cgi.rb中の変数から「開講時期」の値一覧を得る手順

db[“開講時期”]

→[“春学期”, “秋学期”, “春学期”]

db[“開講時期”].uniq

→[“春学期”, “秋学期”]

search-cgi.rb中の変数から「開講時期」の値一覧を得る手順

```
db[“開講時期”].uniq.collect {|j|
“<option>#{j}</option>” }.join(“\n”)
```

→

<option>春学期</option>
<option>秋学期</option>

form-search-cgi.rb前半

```
#!/usr/bin/env ruby
# coding: utf-8
Encoding.default_external = 'utf-8'
require 'csv'
require 'cgi'
c = CGI.new(:accept_charset => "UTF-8")

#データベース読み取り処理は全く同じ
db = CSV.read("syllabus.csv", headers: true)
opts = db["開講時期"].uniq.collect {|j| "<option>#{j}</option>"}.join("\n")

# HTMLヘッダと入力フォーム出力
print <<EOF
Content-type: text/html; charset=utf-8

<!DOCTYPE html>
<html>
<head><title>科目検索</title>
<style type="text/css">
<!--
 tr.head {background: cyan;}
 th {width: 8em;}
 -->
</style>
</head>
<body>
<form method="POST" action="form-search-cgi.rb">
 <h1>科目検索</h1>
 <p>検索条件を入力して下さい。</p>
 <table border="1">
  <tr><td>開講時期</td><td><select name="jiki">
                                        <option value="">------ (指定なし)</option>
EOF

print(opts)     # これで自動生成した option 一覧が出る
```

form-search-cgi.rb後半

```
print <<EOF
</select></td></tr>
 <tr><td>担当教員</td><td><input name="kyoi" size="8"></td></tr>
</table>
<p><input name="ok" type="submit" value="検索">
<input name="cl" type="reset" value="リセット"></p>
</form>
EOF
# 検索パターンはHTMLフォームへの入力値を取得する
term = c["jiki"]
whom = c["kyoi"]

if term>"" or whom>""                    # 検索条件の最低1つが入力されていたら
 if term > ""                            # 開講時期指定に何か文字列を入れたなら
  ptn = Regexp.new(term)    # 文字列を正規表現に変換
  db = db.select {|row| ptn =~ row["開講時期"]}
 end
 if whom > ""                            # 担当教員指定に何か文字列を入れたなら
  ptn = Regexp.new(whom) # 文字列を正規表現に変換
  db = db.select {|row| ptn =~ row["担当教員"]}
 end

 puts("<h2>該当科目一覧</h2>")
 printf("<p>検索語: 開講時期=[%s],  担当教員=[%s]</p>", term, whom)
 puts("<table border=\"1\">")
 db.each {|row|
  printf("<tr class=\"head\"><td colspan=\"2\">%s</td></tr>\n",
      row["科目名"])
  row.each {|key, value|
   printf("<tr><th>%s</th><td>%s</td></tr>\n", key, value)
  }
 }
 puts("</table>")
end
puts("</body>\n</html>")
```

データベースからHTML文を出力する手順

1. データベースを読み込む
2. 自分自身を呼ぶform文を出力する
3. formの値が入力されていたら検索結果を出力

今回の問い

- CGIとは何か？
  - HTML文書内にあるデータ入力窓とそれらを受け渡すスクリプト名を書いておき、ボタンを押すとそのスクリプトに入力値が渡る仕組み
- **データベース検索をCGI化するにはどのようにすれば良いか？**
  - **入力に関わる部分をCGIパラメータから受け取り、出力する部分をHTML形式で書き出す。**
- １つのCGIプログラムで紙芝居のように順次移り変わるWebページを生成するにはどうすれば良いのか？

今回学ぶこと

1. CGI入門
2. 検索主体のCGIプログラムの作成
3. 紙芝居のようにページを進めるCGI

紙芝居のようにページを進めるCGI

1つのCGIプログラムで紙芝居のように順次移り変わるWebページを生成する。

hidden変数の活用

1つのCGIプログラムに複数のWebページ出力を管理させるには、現在の状態を示す変数を**hidden変数**に埋め込んで、次に起動されるCGIプログラムでその値を参照して出力するHTMLを切り替えればよい。

hidden変数の活用

以下の流れで進むWebページ遷移を考える。

1. こんにちは
2. やあやあ
3. さようなら

CGIプログラムの方針

1. 出力すべきページの指定がない場合
   →１ページ目を出力
2. 出力すべきページの指定がある場合
   →そのページを出力

CGIプログラムの方針

ページ番号とそのページでの出力内容
を以下のようなハッシュで定義する。

```
page = {
  "1" => "こんにちは",
  "2" => "やあやあ",
  "3" => "さようなら",
}
```

CGIプログラムの方針

出力すべきページ番号を入力名pageで管理するならHTMLで以下のフォームを記述すれば良い。

```
<form method="POST" action="page2go.rb">
 <input type="hidden" name="page" value="ページ番号">
</form>
```

page2go.rb前半

```
#!/usr/bin/env ruby
# coding: utf-8

require 'cgi'
c = CGI.new(:accept_charset => 'utf-8')

page = {
 "1" => "こんにちは",
 "2" => "やあやあ",
 "3" => "さようなら",
}
p = c["page"]
if p==""
 p="1"                                  # page変数指定がなければ "1" とする
end
name = c["name"]

puts "Content-type: text/html; charset=utf-8

<!DOCTYPE html>
<html>
<head><title>Page #{p}</title></head>
<body>
<h1>Page #{p}</h1>
<form method=\"POST\" action=\"page2go.rb\">"
```

page2go.rb後半

```
# ページ番号による切り替え処理
case p
when "1"
  nextpage = "2"
when "2"
  nextpage = "3"
when "3"
  nextpage = nil                          # 「次のページ」はなし
end
if name > ""                              # nameが定義されていたら呼びかけ文出力
  printf("<p>%sさん!</p>\n", name)
end
printf("<p>%s</p>\n", page[p])          # ページ番号に応じたメッセージ出力

if nextpage
  if name == ""                                    # 名前入力がなければ名前入力フォーム
    puts('<p>お名前: <input type="text" name="name" size="10"></p>')
  else
    # 既に入力されていたら hidden 変数に保存
    printf("<input type=\"hidden\" name=\"name\" value=\"%s\">\n", name)
  end
  printf("<input type=\"hidden\" name=\"page\" value=\"%s\">\n", nextpage)
  puts('<input type="submit" value="次へ">')
end

puts "</form>
</body>
</html>"
```

今回の問い

- CGIとは何か？
  - HTML文書内にあるデータ入力窓とそれらを受け渡すスクリプト名を書いておき、ボタンを押すとそのスクリプトに入力値が渡る仕組み
- データベース検索をCGI化するにはどのようにすれば良いか？
  - 入力に関わる部分をCGIパラメータから受け取り、出力する部分をHTML形式で書き出す。
- **１つのCGIプログラムで紙芝居のように順次移り変わるWebページを生成するにはどうすれば良いのか？**
  - **hidden変数を活用する。**

今回学んだこと

1. CGI入門
2. 検索主体のCGIプログラムの作成
3. 紙芝居のようにページを進めるCGI

課題

GIMP, LibreOffice/Draw, Draw.io, Inkscape, Kritaのいずれかを用いて授業のロゴ画像を作成せよ。

s4基礎プロII(F)の「#07 pf2水 課題 (作画)」に指示どおり書き込む。提出後修正して良い。その場合は「編集」リンクから書き換えること。

締切：**11月22日 (土)**

事前課題

trr「日本国憲法」の練習を行い、s4基礎プロII(F)の「#08 pf2水 事前課題 (trr)」に指示どおり書き込む。提出後修正して良い。その場合は「編集」リンクから書き換えること。

締切：**11月25日 (火)**

今回

第１回　生成AIを効果的に利用した学習法
第２回　メソッド定義と効果的活用（関数的処理）
第３回　メソッド定義と効果的活用（データ集合処理）
第４回　メソッド定義と効果的活用（再帰的アルゴリズム）
第５回　変数のスコープ・クラス設計
第６回　専門演習紹介
第７回　CGIと情報システム（1）
第８回　CGIと情報システム（2）
第９回　チーム課題準備期間（1）
第１０回　チーム課題準備期間（2）・trr試験
第１１回　予選発表
第１２回　代表発表
第１３回　合同成果発表会
第１４回　期末試験