NTNU
Kunnskap for en bedre verden

Leksjon 2: Ingredienser og mye krydder i PHP

Svend Andreas Horgen, IDI Kalvskinnet, NTNU
Lærestoffet er utviklet for emnet IINI3003 Webprogrammering med PHP

Resymé: Programmering er som kokekunst. Uten ingredienser og hjelpemidler kommer en ikke langt. Dersom du som er leser har programmert mye tidligere og kan basisstoffet, anbefaler vi likevel at du ikke hopper over leksjonen. Forhåpentligvis kan mange og enhver snappe opp noen godbiter her og der, eller i det minste litt krydre hverdagen litt.

1. Et lite tilbakeblikk på tekststrenger

Forrige leksjon tok opp samspillet mellom klient og tjener. Variabler, datatyper og tekststrenger ble også gjennomgått. Det er på tide å gå videre gjennom syntaksen til de grunnleggende byggestenene i PHP, og samtidig illustrere hvordan dynamikk kan skapes.

Det er viktig å behandle tekst i websammenheng, og dette kan være en kneik for mange. En kort oppsummering angående tekststrenger skader ikke:

- Tekststrenger kan markeres på flere måter.

```
"Her er en streng i anførselstegn (doble)";
'Her er en streng i fnutter (enkle)';
<<<HVA_SOM_HELST
Det som står her er i en og samme tekststreng
faktisk alt sammen helt til kodeordet fra første linje kommer igjen. <br>
Dette heter forresten <strong>heredoc syntax</strong>, og er nyttig hvis større
elementer av eksisterende HTML-kode med tagger, anførselstegn og/eller fnutter
skal lagres i en tekststreng raskt og effektivt.
HVA_SOM_HELST;
```

- En streng kan ikke stå alene, den må enten lagres i en variabel, skrives ut eller brukes som argument i funksjonskall (leksjon 4 og 5).
- Punktum brukes for å slå sammen tekststrenger. Merk: Du kan fritt bruke punktum i en tekststreng. Den tolkes bare som operator når den står utenfor en tekststreng.

```
$bilde_start = "<img src='";
$filnavn = "mittBilde.jpg";
$bilde_slutt = "'>";
$bilde_slutt_med_hoyde = "' height='200'>";
echo "Her er et bilde: " . $bilde_start . $filnavn . $bilde_slutt;
echo "<br>";
echo "Forrige bilde ble for høyt, her er en mindre utgave: <p>";
echo $bilde_start . $filnavn . $bilde_slutt_med_hoyde;
```

- Variabler kan forekomme inne i tekststrenger, men innholdet vil bare tolkes dersom anførselstegn brukes for å starte strengen.

```
$bra_film = "Shawshank Redemption";
echo "En bra film er <em>$bra_film</em>";
echo "<br>";
echo 'Variabelen $bra_film har innholdet ' . $bra_film;
```

- Tegnet backslash `\` etterfulgt av en spesiell bokstav eller et tegn angir at spesialtegnet skal tolkes som tekst i stedet for å prosesseres som normalt. Særlig `\n` og `\t` er nyttig for formattering av HTML-koden som sendes til nettleseren, siden den kan vises ved å velge ”Vis kildekode”-funksjonaliteten i nettleseren.
- Legg merke til den fine muligheten for å veksle mellom PHP-modus og vanlig HTML-modus. Dersom det kommer store områder med HTML hvor det ikke er krav til dynamikk, er det ingen grunn til å bruke PHP-modus. Dersom du har store bolker med HTML og plutselig skal ha litt PHP, for eksempel bruke en variabel, kan du gjøre slik:

```
<h1>Her er mye html</h1>, bla bla bla
<p>Navnet ditt er: <?php echo $navn ?> så nå vet du det.
```

2. Det som står om kontrollstrukturer i boka

For å kontrollere gangen i hvordan et program skal utføres benyttes kontrollstrukturer. Syntaksen i PHP er nokså lik C og Java, der alt som er plassert innenfor et sett av klammeparenteser etter kontrollstrukturens navn, er en del av den. I eksemplene i dette kurset vil vi ofte bruke kontrollstrukturer, da også gjerne nøstet inn i hverandre.

Med **beslutninger** kan du kontrollere hvorvidt noe skal utføres eller ikke. Livet består av mange valg, og det samme gjør et dataprogram. I ulike situasjoner, under varierende forhold, basert på brukerens inngangsdata – det er mange ganger det vil det være behov for å velge en bestemt handling utfra visse kriterium. I PHP er det mulig å bruke `if` og `switch`.

Boka går gjennom syntaksen til beslutninger i kapittel 3.2. Vær bevisst på eventuell bruk av kortformen som vist i den siste blå rammen i kapittel 3.2 i boka: Den ternære kondisjonelle operatoren. Slik kompakt formulering av if-else er nyttig for å spare plass, men kan være vanskelig å lese for et utrenet øye, og kan gjøre prosessen med å finne feil vanskeligere.

Kjært barn har mange navn. En **løkke**, sløyfe eller loop er en kontrollstruktur som repeterer en kodesnutt basert på en betingelse som evalueres på nytt for hver gang løkken er i ferd med å gjentas. Med løkker oppnås fleksibilitet. Gjennom annen programmering har du sikkert blitt god venn med løkker – de er kraftige i bruk når en først forstår bruken av dem. Når du skal gjenta flere setninger flere ganger og det er liten variasjon i innholdet, bør du med en gang tenke løkker. Løkker gjentar sitt innhold så lenge en betingelse er sann, eller til den avsluttes på annen måte (vha for eksempel kodeordet `exit`).

Det er på samme måte i websammenheng. Kontrollstrukturer passer veldig bra for å:

- Lette arbeidet knyttet til uvikling, særlig i forbindelse med å fylle skjema-elementer med innhold.
- Skape dynamikk og gi sidene et personlig preg. Innholdet kan bestemmes på direkten utfra hva brukeren gjør eller annen informasjon.

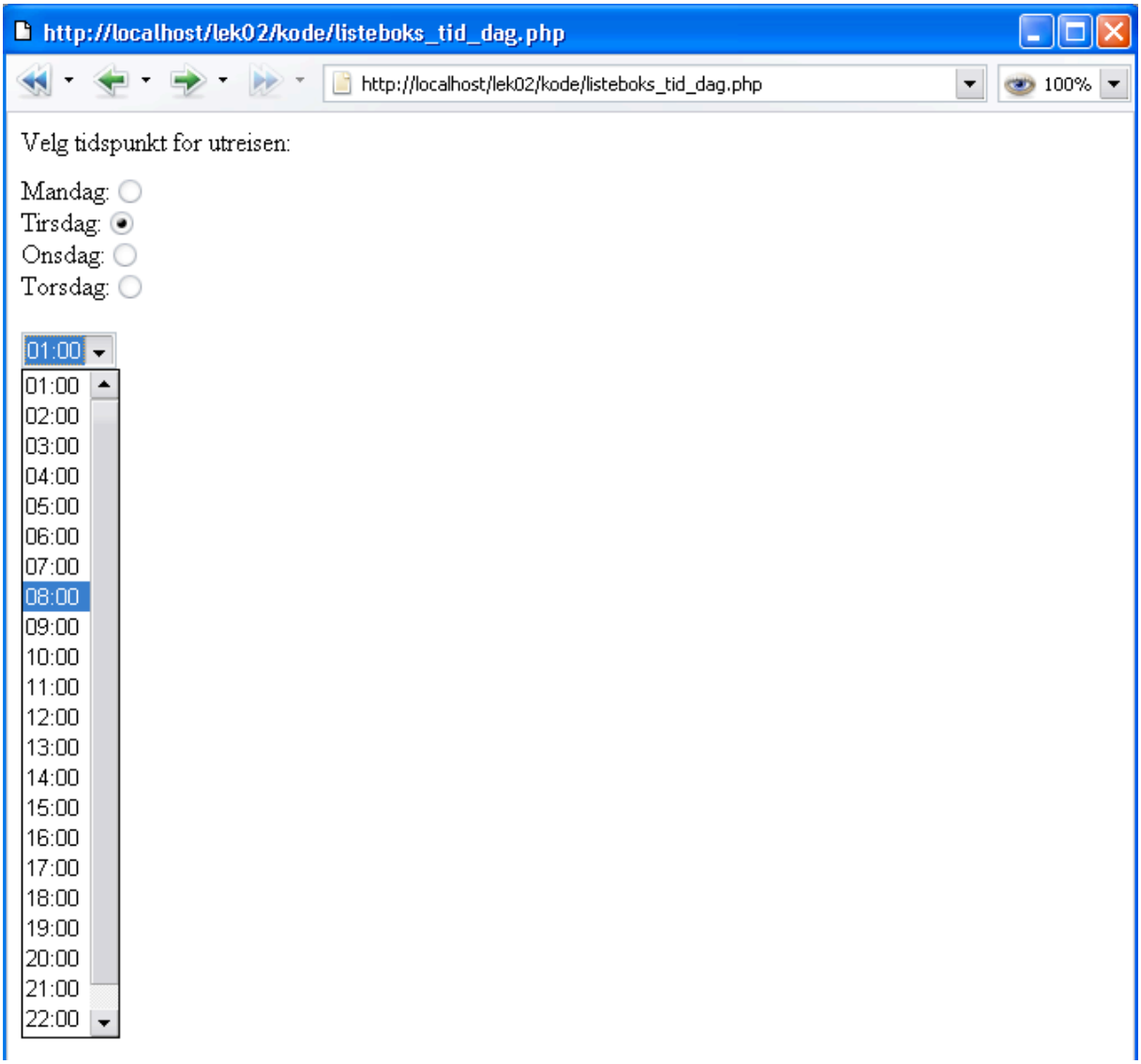
Boka går gjennom et eksempel med hvordan en stor tabell med en fast form for variasjon, kan lages enkelt med en while-løkke. Du kan ta det som en utfordring å lage samme tabell med en for-løkke.

3. Eksempel på bruk av kontrollstrukturer

Skjemabehandling er viktig i websammenheng. Med skjema kan brukeren skrive inn informasjon og trykke på en knapp eller på annen måte sende skjemaet for videre prosessering. Vi kommer inn på *behandling* av skjema i neste leksjon og i stort sett alle etterfølgende leksjoner. Denne gangen skal vi nøye oss med å se på hvordan enkle skjema kan *utformes*, uten at informasjonen som blir skrevet inn blir prosessert. Det å lage grensesnitt er nemlig ofte en stor del av kunsten, og det passer ypperlig å vise hvordan kontrollstrukturer kan brukes til å lage skjema raskt og effektivt med PHP.

3.1. Case: populer en nedtrekksliste med innhold

Tenk deg at du skal fylle en nedtrekksliste med tidspunkt – kanskje for å bruke denne i et bestillingssystem – og at det skal se ut omtrent slik:



Legg først og fremst merke til at siden alle radioknappene har samme navn i name-attributtet, vil det bare være mulig å velge en av dem samtidig i nettleseren. Vi skal se mer på value-attributtet i neste leksjon. Poenget nå er å klare å lage dette brukergrensesnittet på en smart måte.

3.2. HTML-kode for å lage en statisk versjon

Nedtrekkslisten og resten av teksten kan lett lages med HTML-kode slik som vist under.

```
Velg tidspunkt for utreisen:
<form>
      Mandag: <input type="radio" name="dag" value="mandag"><br>
      Tirsdag: <input type="radio" name="dag" value="tirsdag"><br>
      Onsdag: <input type="radio" name="dag" value="onsdag"><br>
      Torsdag: <input type="radio" name="dag" value="torsdag">
      <p>
      <select name="tidspunkt">
      <option value='01:00'>01:00</option>
```

```
      <option value='02:00'>02:00</option>
      <option value='03:00'>03:00</option>
      <option value='04:00'>04:00</option>
      <option value='05:00'>05:00</option>
      <option value='06:00'>06:00</option>
      <option value='07:00'>07:00</option>
      <option value='08:00'>08:00</option>
      <option value='09:00'>09:00</option>
      <option value='10:00'>10:00</option>
      <option value='11:00'>11:00</option>
      <option value='12:00'>12:00</option>
      <option value='13:00'>13:00</option>
      <option value='14:00'>14:00</option>
      <option value='15:00'>15:00</option>
      <option value='16:00'>16:00</option>
      <option value='17:00'>17:00</option>
      <option value='18:00'>18:00</option>
      <option value='19:00'>19:00</option>
      <option value='20:00'>20:00</option>
      <option value='21:00'>21:00</option>
      <option value='22:00'>22:00</option>
      <option value='23:00'>23:00</option>
      </select>
</form>
```

Synes du det er spesielt enkelt og greit å lage denne koden? Hva skjer i det øyeblikk du vil legge inn hver halve time også? Det *er* tidkrevende å hardkode slike lister i vanlig HTML.

3.3. Nedtrekksliste laget med PHP

Her er en modifisert, mye ”smartere” løsning som tar utgangspunkt i mønsteret vi har sett til nå. Mønsteret er helt tydelig, i og med at timene øker med nøyaktig én for hver gang. Ved å bruke en for-løkke kan PHP gjøre jobben for oss:

```
Velg tidspunkt for utreisen:
<form>
      Mandag: <input type="radio" name="dag" value="mandag"><br>
      Tirsdag: <input type="radio" name="dag" value="tirsdag"><br>
      Onsdag: <input type="radio" name="dag" value="onsdag"><br>
      Torsdag: <input type="radio" name="dag" value="torsdag">
      <p>

      <select name="tidspunkt">
            <?php
            for ($i=1; $i<24; $i++) {
                  echo "\t<option value='";
                  if ($i<10) $nuller = "0";
                  else $nuller = "";
                  $tid = $nuller . $i . ":00";
                  echo "$tid'>$tid</option>\n";
            }//for
```

```
            ?>
      </select>
</form>
```

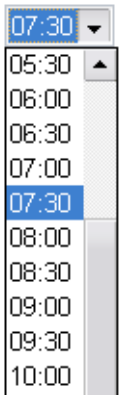
Resultatet blir det samme som i eksempelet med den hardkodede HTML-koden. Dette eksempelet illustrerer praktisk bruk av kontrollstrukturer i forbindelse med webutvikling. If-setningen som avgjør hvorvidt det skal settes inn en ledende 0-er eller ikke, gjør at resultatet blir penere (gitt at en synes det er penere å lese ”03:00” enn ”3:00” i nedtrekkslisten).

Hvis nå hver halve time skal lures inn i tillegg, er det bare å legge til tre PHP-setninger og endre det ene tallet til 30 i stedet for 00 i for-løkken.

```
<?php
for ($i=1; $i<24; $i++) {
      //lager starten av en option-tag, og finner ut
      //om det skal være ledende nullere
      echo "\t<option value='";
      if ($i<10) $nuller = "0";
      else $nuller = "";

      //lager tidspunkt med :00 på slutten
      $tid = $nuller . $i . ":00";
      echo "$tid'>$tid</option>\n";

      //lager ny option-tag, nå med tidspunkt :30 på slutten
      echo "\t<option value='";
      $tid = $nuller . $i . ":30";
      echo "$tid'>$tid</option>\n";
}//for
?>
```



Hva nå om du bestemmer deg for at hver søndag skal bare timene fra 09:00 til 17:00 vises? For-løkken må endres litt, slik at den teller fra start til slutt, i stedet for fra 1 til 24, og deretter må en if-test avgjøre om det er søndag i dag:

```
$dagen_i_dag = date("l");  //liten L, ikke ett-tall
if ($dagen_i_dag == "Sunday") {
      $start = 9;
      $slutt = 17;
}
else {
```

```
        $start = 1;
        $slutt = 23;
}
for ($i=$start; $i<=$slutt; $i++) {
            //akkurat samme kode som før her
}
```

Som du ser kan PHP brukes for å lage skjema effektivt, men også for å skape dynamikk du ellers ikke ville klart. Det er fleksibelt å lage kode med PHP, men bruk av PHP skaper også et større behov for å kommentere med tanke på fremtidig utvidelse. Venn deg også til å bruke kommentarer i koden fra starten av!

4. Assosiative matriser (associative arrays)

En variabel kan inneholde informasjon, men bare en ting om gangen. En matrise (array på engelsk, også kalt tabell på norsk) kan lagre mange ting samtidig. En matrise er derfor en samling av informasjon. En god grunn til at matriser ofte er å foretrekke fremfor en rekke frittstående variabler, er at det er enkelt å gjennomløpe hele eller deler av matrisen ved å bruke løkker. I tillegg kan vi gjøre kraftige operasjoner på en matrise, så som å sortere dens innhold, skrive ut visse deler av den og behandle dens data, for å nevne noe.

4.1. Matriser i PHP

Matriser i PHP er egentlig en litt spesiell datatype som knytter de ulike elementenes verdier til tilhørende nøkler. Vi kommer tilbake til mange praktiske eksempler med matriser gjennom kurset, men utforsker først idéen med assosiative matriser.

Du kan sammenlikne en tradisjonell endimensjonal matrise (fra annen programmering) med en tabell som har én kolonne og en rekke rader. Noen vil kalle dette for nettopp bare *tabell*, og bruker ordet *matrise* når det er snakk om flere kolonner og flere rader. Vi bruker ordet matrise i PHP, oversatt fra det engelske ordet *array*. Krysningen mellom en rad og en kolonne kalles et *element*. I elementene kan det lagres data.

Før vi kan hente ut data fra en matrise må den opprettes. I PHP er det enkleste å bruke PHP-konstruksjonen `array` til dette formålet:

```
$minMatrise = array(1,2,-8,40);
```

Setningen fører til at matrisen med navn `$minMatrise` blir fylt med verdiene 1, 2, -8 og 40. Følgende syntaks kan alternativt benyttes for å lage samme matrise:

```
$minMatrise[] = 1;
$minMatrise[] = 2;
$minMatrise[] = -8;
$minMatrise[] = 40;
```

Den siste teknikken vil ofte være nødvendig, for eksempel når en verdi skal legges til på slutten av en eksisterende matrise. Den kan brukes selv om matrisen er opprettet med `array`. Etter at setningen under har kjørt, vil matrisen ha fem elementer:

```
$minMatrise[] = -9;
```

En matrise kan bestå av mange elementer. Hvert element har en *nøkkel* (eller indeks som vi ofte sier) og en tilhørende *verdi*. Verdien kan være enten tekst, et tall, eller en ny matrise. Det spesielle med matriser i PHP er at:

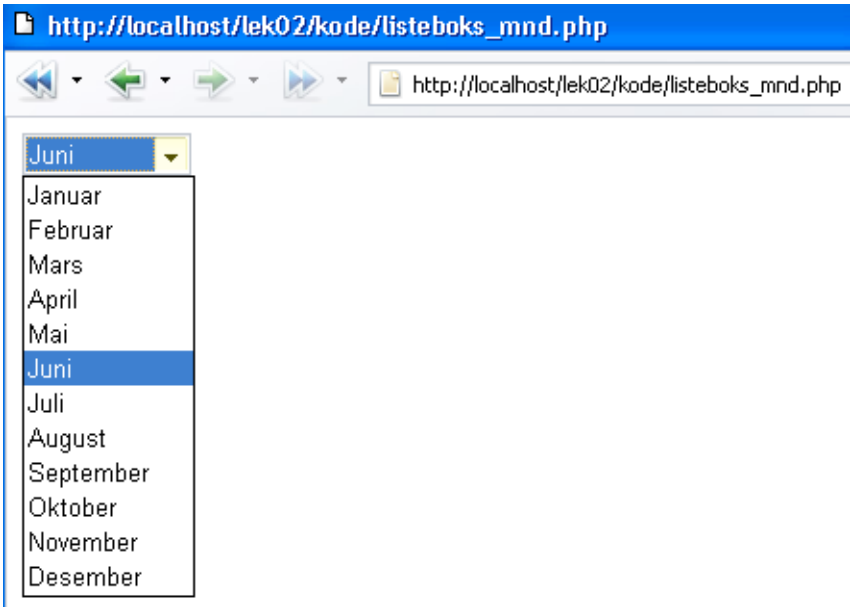
- Ulike elementer kan ha ulik datatype. Dette henger sammen med at variabler ikke skal deklareres i PHP. Du vil trolig sette pris på denne egenskapen etter hvert som du ser ulike bruksområder.
- Nøkkelen kan være enten numerisk, eller tekstlig, og det er dermed opp til deg som utvikler å bestemme innhold i både nøkkel og verdi for hvert enkelt element. Dette gir store muligheter når det kommer til enkel databehandling. Figur 4.1 og 4.2 i boka illustrerer denne viktige forskjellen på ikke-assosiativt og assosiativt.

Nøklene må være unike. Hvordan skulle ellers PHP vite hvilket element som skal hentes fram?

Det kan være vanskelig å forstå hvordan assosiative matriser fungerer, fordelen med slike, og hvordan de best kan brukes. Du kan se en demonstrasjonsfilm (video fra ressurssiden) som forklarer mye av hemmeligheten bak assosiative matriser for å få en bedre forståelse av hva som skjer.

4.2. Løkker + matriser = sant

Vi kan bruke løkker til å hente ut data fra en matrise på en rask og effektiv måte. Når vi skal opprette en matrise kan det ofte være lite hensiktsmessig å bruke den standard nullbaserte indeksen, og vi står da fritt til å starte på for eksempel indeks 1 eller noe annet. Neste eksempel viser hvordan vi først legger inn alle årets måneder i matrisen `$mnd` og deretter skriver ut alle sammen i en nedtrekksliste. Målet er et skjermbilde som ser slik ut:



Nedtrekkslisten/listeboksen kan lages med vanlig HTML, og den vil i så fall se slik ut:

```
<form>
      <select name="tidspunkt">
      <option value='1'>Januar</option>
      <option value='2'>Februar</option>
```

```
      <option value='3'>Mars</option>
      <option value='4'>April</option>
      <option value='5'>Mai</option>
      <option value='6'>Juni</option>
      <option value='7'>Juli</option>
      <option value='8'>August</option>
      <option value='9'>September</option>
      <option value='10'>Oktober</option>
      <option value='11'>November</option>
      <option value='12'>Desember</option>
      </select>
</form>
```

Selv om vi har å gjøre med tekst, er det mulig å bruke PHP og en løkke til å lage listen for oss.

```
<?php
      //lager en matrise med månedsnumre og navn
      $mnd = array (1 => 'Januar',
                  'Februar',
                  'Mars',
                  'April',
                  'Mai',
                  'Juni',
                  'Juli',
                  'August',
                  'September',
                  'Oktober',
                  'November',
                  'Desember'
            );  //slutt på matrisen her. Siste element har nøkkelen 12.
?>
<form>
<select name="tidspunkt">
<?php
foreach ($mnd as $maanedsnummer => $maanedsnavn) {
      //lager option-tag med neste månedsnavn.
      //Nummer blir value-attributt
      echo "\t<option value='$maanedsnummer'>$maanedsnavn</option>\n";
}//foreach - ferdig med å gjennomgå hele matrisen
?>
</select>
</form>
```

Du synes kanskje at det er vel så tungvindt å lage en matrise som å hardkode? Tja, akkurat i dette eksempelet ble det faktisk det, men i mange andre tilfeller er det mer effektivt og ikke minst bedre kodepraksis. Bruk av matriser blir spesielt nyttig i forbindelse med gjenbruk av kode (mer om inkludering i leksjon 4 og kapittel 5 i boka) eller når for eksempel flere nedtrekkslister med de samme månedene skal lages på samme side.

For lesbarheten sin skyld har vi valgt å la hvert element stå på en ny linje i stedet for å ha dem fortløpende på samme linje. Tolkeren bryr seg ikke om hvor mange linjeskift vi måtte velge å bruke, bare vi husker komma mellom hvert element. Legg merke til at vi bare har angitt

nøkkel for det første elementet. Siden vi ikke har sagt noe spesielt om nøklene til de andre elementene, blir hvert element indeksert automatisk med ett høyere nummer enn forrige tallbaserte nøkkel som ble brukt. Siden vi bruker 1 om det første elementet, vil verdien ”Desember” få indeksen 12, og ikke 11.

5. Oppsummering

Vi skal komme mer tilbake til løkker og matriser i de kommende leksjonene, særlig i forbindelse med skjemabehandling. De neste leksjonene vil ta for seg viktige deler av det vi kjenner som webprogrammering. Håper at du har lyst på mer, for vi har bare så vidt sett starten av hva som er mulig å få til med webprogrammering.

Synes du denne leksjonen var tung å lese kan det skyldes at du mangler basiskunnskapene (eller mangler boka). Bruk i så fall litt tid på å tilegne deg nødvendig kunnskap og les deretter denne leksjonen om igjen. Spør i diskusjonsforumet om du lurer på noe.

Med de beste ønsker om et godt semester i PHP sitt interessante kjøkken!