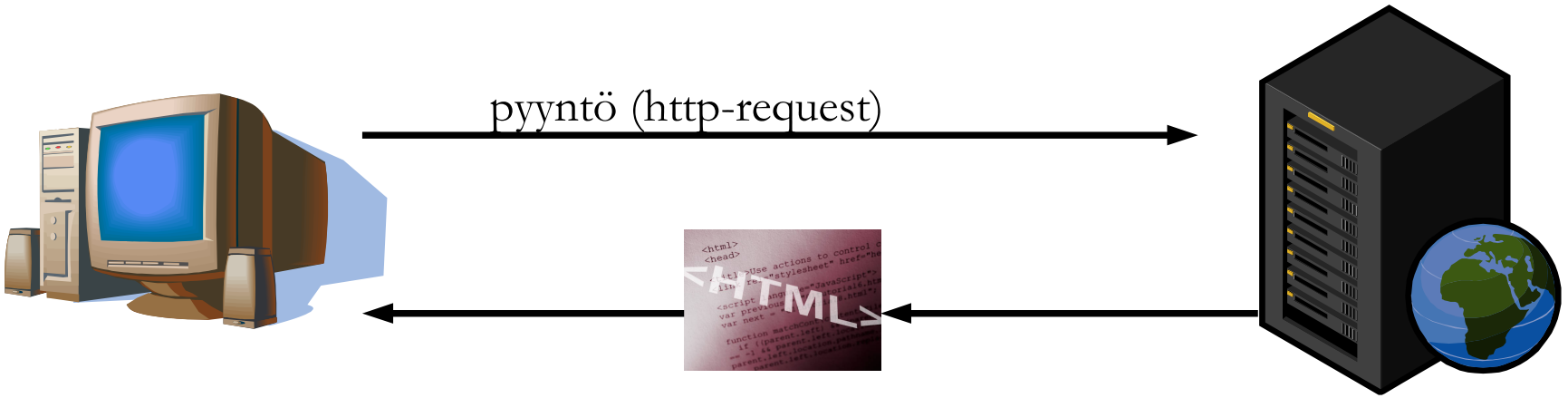
Palvelinpuolen ohjelmointi

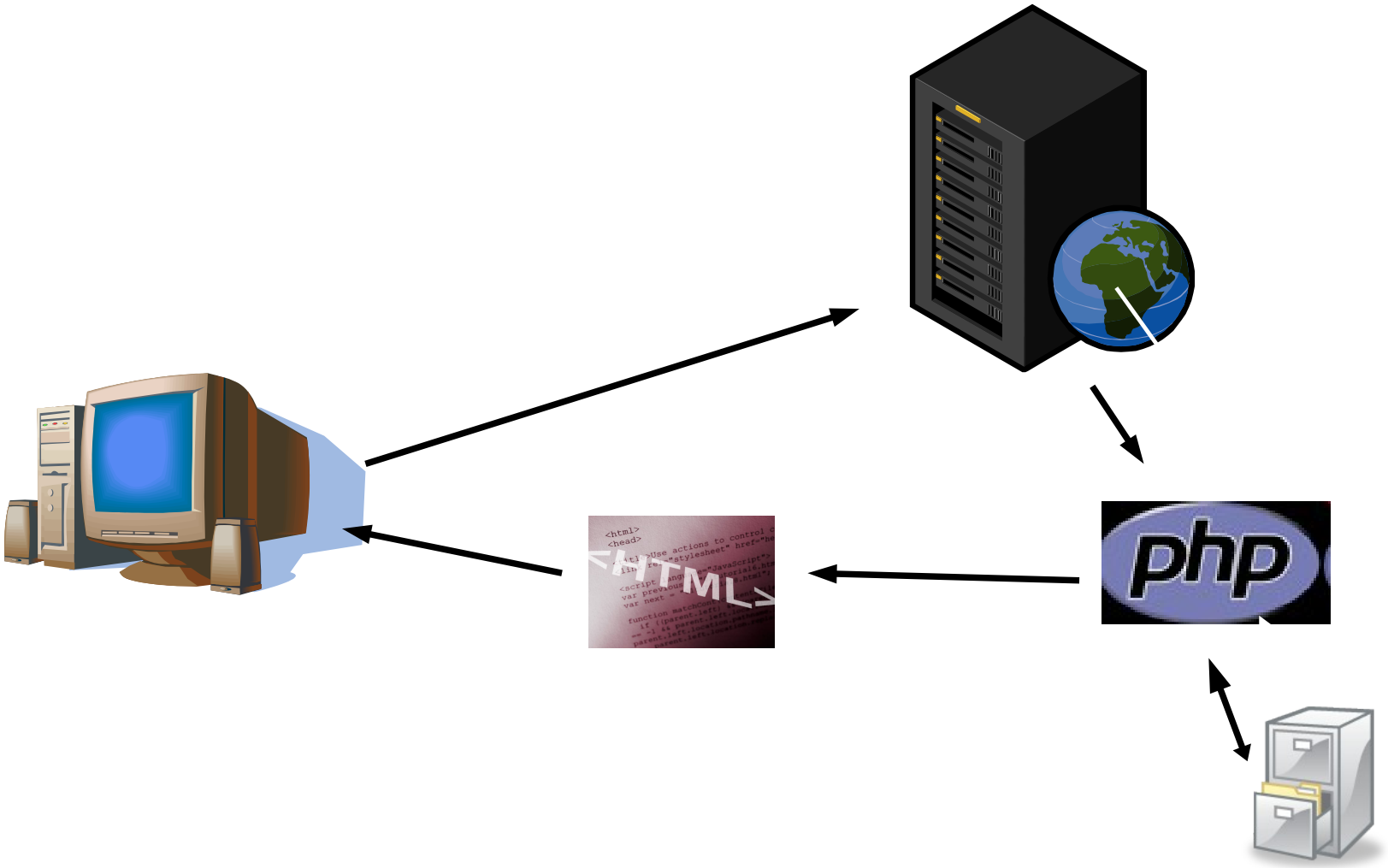
Staattiset ja dynaamiset web-sivut

- Staattiset web-sivut ovat valmiissa muodossaan palvelimella (tai paikallisesti omalla koneella).
- Javascript mahdollistaa paikalliset dynaamiset sivut
- Palvelinpuolen dynaamiset web-sivut *muodostetaan* palvelinpäässä selaimen pyytäessä sivua.
- *Selaimen* ja *selailijan* kannalta eroa ei siis ole.

Staattiset sivut



Dynaamiset sivut



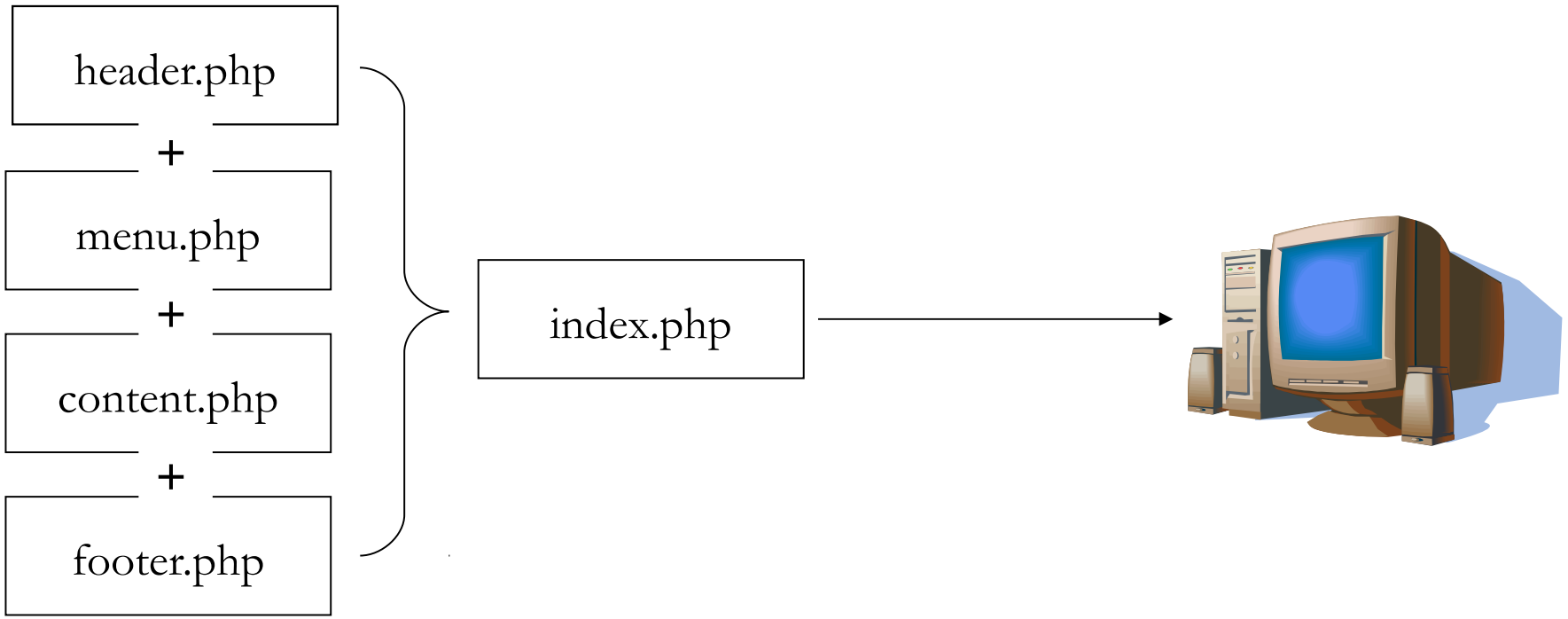
Dynaamisuus

- Dynaamisuus siis tarkoittaa, että PHP (tai jokin muu prosessori) ”tulostaa” sivun sitä pyydettäessä, ja syöttää valmiin HTML-sivun takaisin selaimelle.

- Koska PHP *suoritetaan palvelinpäässä*, loppukäyttäjä näkee ainoastaan valmiin sivun (tai sen lähdekoodin), eikä varsinaista PHP-koodia.

Mitä hyötyä dynaamisuudesta?

1. Sivun muodostaminen erillisistä komponenteista:



Mitä hyötyä dynaamisuudesta (2)

2. Sivun muodostaminen käyttäjän syötteen mukaan

- Sisältö voidaan hakea tiedostosta, tietokannasta tai generoida kokonaisuudessaan alusta alkaen. Yleensä haettavaan (tai muodostettavaan) tietoon vaikuttaa jokin käyttäjän valinta.

- Mahdollistaa myös Web 2.0 palvelut.

PHP

- PHP: Hypertext Preprocessor
- Ensimmäinen versio vuonna 1995
- Nykyinen versio 5.2.8
- Käytetään *yleensä* palvelinpään skriptikielenä.
- Lisätietoja, tutoriaaleja ja ohjeita: http://www.php.net

PHP vs. Python – eräitä eroja

- PHP:ssä ohjelmarivit päättyvät puolipisteeseen (pois lukien lohkojen alku- ja loppumerkit).
- PHP:ssä lohkot merkitään aaltosulkeilla {}, Pythonissa sisentämällä
- PHP:ssä muuttujien nimet aloitetaan aina dollarimerkillä.

PHP vs. Python - esimerkki

```
def SYT():
    a = 408
    b = 112
    c = 0
    while true:
        if b == 0:
            print "SYT:",a
            break

        c = a % b
        a = b
        b = c
```

```
function SYT(){
    $a = 408;
    $b = 112;
    $c = 0;
    while (true){
        if ($b == 0){
            echo 'SYT: '.$a;
            break;
        }
        $c = $a % $b;
        $a = $b;
        $b = $c;
    }
}
```

Merkitseminen

- PHP-koodi merkitään <?php ?>–tagin avulla.
- Huomaa, että koodi tulee *tagin sisään*, eikä tagien väliin.

```
<?php
  echo 'Terve kaikki!';
?>
```

Merkitseminen (2)

- PHP-koodia voidaan kirjoittaa *mihin tahansa* HTML-tiedostossa; koska koodin paikalle tulee selaimeen sen (mahdollisesti) tulostama merkkaus, PHP-koodi ei sinänsä vaikuta sivun validiuteen.

- Huomaa, että <?php ?> -tagia on käytettävä myös silloin, kun tiedostossa on pelkkää PHP-koodia!

Tuloste selaimeen

- PHP tarjoaa muutamia erilaisia funktioita selaimeen tulostamista varten: echo, print, printf jne.

- Pääasiassa kurssilla käytetään kuitenkin echo-komentoa.

echo - syntaksi

- echo-komennon syntaksi:

```
echo <tulostettava_merkkijono>;
```

esim.

```
echo ’Moi kaikki!’;
echo ’<li>Lista-alkio</li>’;
```

Tuloste HTML-sivulla

- Niinkuin aikaisemmin mainittiin, näkyy echo-komennon tuloste tekstinä selaimessa:

esim.php palvelimella

```
<body>
   <p>
   <?php
      echo 'Moi kaikki';
   ?>
   </p>
...
```

esim.php selaimessa

```
<body>
   <p>
   Moi kaikki!
   </p>
...
```

Tuloste HTML-sivulle (2)

- Huomaa, että komennolla voidaan tulostaa useampia rivejä kerralla:

```
<?php
  echo'
     <table>
       <tr><td>Solu 1</td><td>Solu 2</td></tr>
       <tr colspan=”2”>Isompi solu 3</td></tr>
     </table>';
?>
```

echo – syntaksi (2)

- Merkkijonoliteraalit merkitään PHP-ssa lainaus- tai puolilainausmerkkeihin. Numerot merkitään sellaisenaan, desimaalierottimena käytetään pilkun sijasta pistettä.

```
echo ’Tämä on tekstiä!’;
echo ”Tämäkin on tekstiä”;
echo 1500;
echo 25.05;
```

Katenointi

- Merkkijonoja voidaan yhdistää toisten merkkijonojen ja numeroiden kanssa katenoimalla ne. PHP:ssä katenointimerkkinä käytetään pistettä katenoitavasta tiedosta riippumatta:

```
echo ’Moro ’ . ’kaikki!’;
echo ’Hinta on ’ . 500 . ’&euro’;
```

Katenointi (2)

- Katenointiin voidaan yhdistää myös laskutoimituksia. Toimivuuden varmistamiseksi laskutoimitukset kannattaa yleensä merkitä sulkeiden sisälle:

```
echo ’Tulos on ’ . (2 + 3);
echo ’Neljän kertoma on ’ . (4*3*2*1);
echo ’Vuorokaudessa on ’ . (24 * 60) . ’
  minuuttia’;
echo ’<td style=”width:’ . (10*5+5) .’px;”> ’;
```

Laskuoperaattorit

+	**Yhteenlasku**
-	**Vähennyslasku**
*	**Kertolasku**
/	**Jakolasku**
%	**Jakojäännös**
++	**Arvon kasvatus yhdellä**
--	**Arvon vähennys yhdellä**

Muuttujat

- Koska PHP on heikosti tyypitetty ohjelmointikieli (kuten mm. Python), ei muuttujille anneta tyyppiä niitä määriteltäessä, vaan tulkki päättelee sen itse.

- Muuttujan nimi alkaa $-merkillä ja saa sisältää kirjaimia, numeroita ja alaviivaa. Välilyönnit eivät ole sallittuja. Kirjainkoko on merkitsevä, luku ei siis ole sama muuttuja kuin Luku.

Muuttujan arvon asetus

- Muuttujan arvo asetetaan = -operaattorin avulla. Ensimmäinen asetus alustaa muuttujan ja seuraavat muuttavat sen arvoa.

```
$nimi = ”Esko Esimerkki”;
$rivinvaihto = ”<br>”;
$luku = 100;
$luku = 200; // Arvon muutos
$isompi_luku = 1000000;
```

Muuttujan arvon asetus (2)

- Arvoa asetettaessa voidaan luonnollisesti käyttää sekä laskutoimituksia että merkkijonojen katenointia:

```
$summa = 25 + 33;
$summa2 = $summa + 40;
$nimi = ”Keijo” . ” ” . ”Keksitty”;
$palsta = ”<p>” . $nimi . ”</p>”;
$lauseke = ”2 + 5 = ” . (2+5);
```

Kontrollirakenteet

- PHP:ssa voidaan käyttää suorituksen ohjaukseen Pythonista tuttuja kontrollirakenteita: ehtolauseita ja silmukoita.
- if- ja while-lauseet ovat syntaksiltaan jokseenkin yhteneväisiä Python-vastineidensa kanssa, for-lause taas eroaa jonkin verran.

Ehtolauseet

- Ehtolauseella voidaan määrätä jokin käsky tai käskylohko suoritettavaksi silloin, kun jokin ehto on tosi
- PHP:ssa ehtolause on muotoa

```
if (ehto){
    suoritettava käsky tai lohko
}
```

Ehtolauseet (2)

- Ehtona voidaan käyttää joko totuusarvotyyppistä muuttuja (arvo true tai false) tai kahden lausekkeen vertailua jotain vertailuoperaattoria käyttäen.
- Huomaa, ettei totuusarvon ympärille kirjoiteta lainausmerkkejä:

```
$totuus = false;

$kirjauduttu = true;
```

Esimerkki:

- Tulostetaan merkkijono "moi", mikäli muuttujan $totuus arvo on tosi. *Huomaa, että ehto kirjoitetaan sulkujen sisälle!*

```
if ($totuus){
    echo 'moi';
}
```

Vertailuoperaattorit

==	Yhtäsuuruus
!=	Erisuuruus
!	Epätosi
<	Pienempi kuin
>	Suurempi kuin
<=	Pienempi tai yhtäsuuri kuin
>=	Suurempi tai yhtäsuuri kuin

Vertailuoperaattorit - esimerkkejä

$a == 4	Tosi, jos $a on yhtäsuuri kuin 4
$b == $c	Tosi, jos $b on yhtäsuuri (tai sama) kuin $c
$x != $y	Tosi, jos $x on erisuuri kuin $y
!$totuus	Tosi, jos $totuus on epätosi (false)
$m > 3	Tosi, jos $m on suurempi kuin 3
$n < 5	Tosi, jos $m on pienempi kuin 5
$t >= 11	Tosi, jos $t on suurempi TAI yhtäsuuri kuin 11
$c == 'moi'	Tosi, jos $c:n arvo on merkkijono *moi*

Esimerkkejä

- Tulosta "suurempi on", jos $luku on suurempi kuin 10:

```
if ($luku > 10){
    echo 'suurempi on';
}
```

- Katenoi merkkijonoon toinen merkkijono, jos sen alkuperäinen arvo on "Esko":

```
if ($nimi == "Esko"){
    $nimi = $nimi + " Esimerkki";
}
```

Esimerkkejä (2)

- Tulosta otsikko, mikäli $valmis on epätosi:

```
if (!$valmis){
    echo '<h2>Tuotetiedot</h2>';
}
```

- Lisää muuttujan $summa arvoon muuttujien $a ja $b arvot, mikäli niiden summa on suurempi tai yhtäsuuri kuin 10:

```
if (($a + $b) > 10){
    $summa = $summa + $a + $b;
}
```

Esimerkkejä (2)

- Tulosta otsikko, mikäli $valmis on epätosi:

```
if (!$valmis){
    echo '<h2>Tuotetiedot</h2>';
}
```

- Lisää muuttujan $summa arvoon muuttujien $a ja $b arvot, mikäli niiden summa on suurempi tai yhtäsuuri kuin 10:

```
if (($a + $b) > 10){
    $summa = $summa + $a + $b;
}
```

Ehtolause ja vaihtoehtoinen haara

- else-komennolla voidaan määritellä ehtolauseelle vaihtoehtoinen haara.
- Tämä vaihtoehtoinen haara suoritetaan silloin, kun alkuperäinen ehto on epätosi.

if – else -syntaksi

```
if (ehto){
    lohko 1
}
else {
    lohko 2
}
```

- Lohko 1 suoritetaan mikäli ehto on tosi, ja lohko 2 mikäli ehto on epätosi.

Esimerkkejä

- Tulosta muuttujista $a ja $b suurempi:

```
if ($a > $b){
    echo $a;
}else {
    echo $b;
}
```

Esimerkkejä (2)

- Esimerkissä asetetaan muuttujaan $pass arvo sen mukaan, onko muuttujassa $user merkkijono 'admin' vai ei.

```
if ($user == "admin){
    $pass = true;
}else{
    $pass = false;
}
```

Sisäkkäiset lohkot

- If-lohkoja voidaan luonnollisesti kirjoittaa myös sisäkkäin. Esimerkissä tulostetaan kolmesta luvusta suurin:

```
if ($a > $b){
        if ($a > $c){
                echo $a; // a suurempi kuin b JA c
        }else{
                echo $c; // a suurempi kuin b ja c suurempi kuin a
        }
}else{
        if ($b > $c){ // a pienempi kuin b ja b suurempi kuin c
                echo $b;
        }else{
                echo $c; // a pienempi kuin b ja b pienempi kuin c
        }
}
```

Toistolause

- Toistolauseen avulla voidaan suorittaa lohkoa niin kauan, kun ehto on voimassa.
- PHP:n toistolauseen syntaksi vastaa pitkälti Pythonin vastaavaa:

```
while (ehto){
    toistettava lohko
}
```

Esimerkkejä

- Esimerkissä tulostetaan lukuja niin kauan, kun muuttujan $a arvo on pienempi kuin 10:

```
while ($a < 10){
    echo $a . '<br>';
    $a++; // Kasvata $a:n arvoa yhdellä.
}
```

Esimerkkejä (2)

- Tulosta listaan seitsemän alkiota:

```
$luku = 1;
echo '<ul>';
while ($luku <= 7){
    echo '<li>Alkio ' . $luku . '</li>';
    $luku++;
}
echo '</ul>';
```

Esimerkkejä (3)

- Ehto- ja toistolauseiden yhdistäminen: Tulosta kaikki kolmella jaolliset luvut välillä 0...20 suurimmasta pienimpään:

```
$n = 20;
while ($n >= 0){
    if ($n % 3 == 0){
        echo $n . '<br>';
    }
    $n = $n - 1;
}
```

Toistolause: for

- Tiettyyn rajaan asti toistettava lohko voidaan toteuttaa myös for-lauseella.
- for-lause vastaa periaatteessa Pythonin for ... in range –rakennetta, mutta on monipuolisempi.

for-lause: syntaksi

```
for (asetus ; ehto ; muutos) {
    toistettava lohko
}
```

- asetus: alustetaan silmukkamuuttuja
- ehto: lohkoa suoritetaan niin kauan kun ehto ON voimassa
- muutos: jokaisen kierroksen päätteeksi muuttujaan tehtävä muutos.

Esimerkkejä

- for-silmukka, joka tulostaa luvut 1...10

```
for ($i=1; $i<=10; $i++){
    echo $i;
}
```

- asetus: alustetaan muuttujan $i arvoksi 1
- ehto: lohkoa suoritetaan niin kauan kun muuttujan $i arvo on pienempi tai yhtäsuuri kuin 10
- muutos: jokaisen kierroksen päätteeksi muuttujan $i arvoa kasvatetaan yhdellä.

Esimerkkejä (2)

- Tulosta teksti jatkuvasti pienenevällä fontilla:

```
for ($i = 36; $i>=12; $i = $i - 3){
    echo '<p style="font-size:' . $i . 'px">';
    echo 'Terve kaikki!';
    echo '</p>';
}
```

while- ja for-lauseen erot

- Huomaa, että for-silmukassa silmukkamuuttujan arvoa ei tarvitse (eikä yleensä kannata) erikseen muuttaa silmukan sisällä; for-lauseen muutoslause riittää.
- while-silmukassa muutos on kuitenkin tehtävä erikseen.

while- ja for-lauseen erot

- while-silmukassa ehto voi kuitenkin olla monipuolisempi.
- Yleisesti: käytä for-silmukkaa, kun tarvitset lukusarjaa tietystä alkuarvosta tiettyyn loppuarvoon säännöllisin välein. Muissa tapauksissa while-silmukka on luultavasti kätevämpi.

Sisäkkäiset silmukat

- Silmukoitakin voidaan kirjoittaa sisäkkäin:

```
for ($i=2; $i < 6; $i++){
    for ($j=1; $j < 5; $j++){
        echo $i . '+' . $j . '=' . ($i + $j);
    }
}
```

Syötteiden lukeminen

- Toistaiseksi läpikäydyt ohjelmat ovat näennäisestä dynaamisuudestaan huolimatta kuitenkin tavallaan staattisia: tuloste on joka kerralla sama.
- Vuorovaikutuksen saavuttamiseksi tarvitaan keinoja syötteiden lukemiseksi.

Keinot syötteiden lukemiseen

- Koska PHP-ohjelmat suoritetaan palvelinpäässä, ei Pythonista tuttua input-dialogia voida käyttää syötteiden lukemiseen.
- PHP tarjoaa kuitenkin kaksi mahdollisuutta arvojen välitykseen: osoiteriviparametrit ja lomakkeet.

Osoiteriviparametrit

- Osoiteriville voidaan lisätä parametreja sivun varsinaisen osoitteen perään. Parametrit näkyvät käsittelevässä PHP-ohjelmassa muuttujina.
- Syntaksi on seuraava:

  www.osoite.com/tiedosto.php?
  parametrin_nimi=arvo

Osoiteriviparametrit (2)

- Esimerkiksi:

  http://127.0.0.1/ohjelma.php?luku=100

  http://localhost/oma.php?summa=99

  http://www.example.com/my.php?id=user

Osoiteriviparametrit (3)

- Parametreja voidaan välittää useampia erottamalla ne &-merkillä:

http://127.0.0.1/testi.php?eka=5&toka=11

http://www.example.com?a=moi&b=kaikki

http://localhost/laske.php?x=5&y=8&z=23

Parametrien lukeminen PHP:lla

- PHP-ohjelmassa voidaan lukea välitetyt parametrit hyödyntämällä globaalia taulukkomuuttujaa $_GET
- Taulukko sisältää kaikki osoiteriviltä välitetyt parametrit.

Parametrien haku taulukosta

- Haluttuun parametriin viitataan taulukossa sen nimellä: $_GET['parametrin_nimi'].
- Huomaa, että nimi on hyvä kirjoittaa puolilainausmerkkeihin (eli hipsukoihin).

Esimerkkejä

- Eli asetetaan komentoriviparametrin "a" arvoksi 20 ja tulostetaan arvo PHP-ohjelmassa:

  komentorivi:

  http://127.0.0.1/oma.php?a=20

  oma.php:

```
<?php
    echo $_GET['a'];
?>
```

-

Esimerkkejä (2)

- Lasketaan yhteen annettujen parametrien arvot uuteen muuttujaan $summa:

  komentorivi:

  http://127.0.0.1/laske.php?eka=10&toka=25

  laske.php:

```
<?php
    $a = $_GET['eka'];
    $b = $_GET['toka'];
    $summa = $a + $b;
?>
```

Esimerkkejä (3)

- Tulostetaan parametrina annettu määrä lukuja:

komentorivi:

http://127.0.0.1/lukuja_listassa.php?maara=8

lukuja_listassa.php:

```
<?php
    echo '<ul>';
    for ($i=0; $i<$_GET['maara']; $i++){
        echo '<li>' . $i . '</li>';
    }
    echo '</ul>';
?>
```

Parametrit käytännössä

- Yleensä ei voida olettaa, että käyttäjä kirjoittaisi parametrit arvoineen komentoriville, vaan ne tarjotaan linkkeinä toiselta (tai samalta) sivulta.
- Huomaa kuitenkin, että arvot ovat käyttäjän muuteltavissa! Sen takia arvot on yleensä aina tarkistettava ohjelmassa!

Parametrit linkeissä

- Esim. kolme linkkiä, joista jokainen välittää eri arvon ohjelmalle tulosta.php:

```
<a href="tulosta.php?arvo=1">eka</a>
<a href="tulosta.php?arvo=2">toka</a>
<a href="tulosta.php?arvo=3">kolmas</a>
```

tulosta.php:

```
echo 'Saatiin arvo ' . $_GET['arvo'];
```

Parametrit linkeissä (2)

- Valikko, joka välittää sisältösivun id-numeron ohjelmalle get_content.php:

```
<ul id="menu">
    <li><a href="get_content.php?id=1">etusivu</a></li>
    <li><a href="get_content.php?id=2">uutiset</a></li>
    <li><a href="get_content.php?id=3">tuotteet</a></li>
    <li><a href="get_content.php?id=4">yhteystiedot</a></li>
</ul>
```

Parametrit linkeissä (3)

- Tiedosto get_content.php

...Sivun alku tässä ...

```
<?php
    if (isset($_GET['id'])){
        $id = $_GET['id'];
        if ($id >= 1 && $id <= 4){
            include ('page' . $id . '.php');
        }
    }
?>
```

... Sivun loppu tässä ...

Parametrien tarkistus

- Ennen käsittelyä pitää kuitenkin yleensä tarkistaa, että

  1) Parametri on asetettu JA

  2) Parametrin arvo asettuu halutulle välille

Parametrin olemassaolo

- Parametrin olemassaolo tarkistetaan isset – funktiolla
- Funktion palauttaa arvon true, mikäli annettu muuttuja on olemassa
- Esim:

```
if (isset($_GET['id'])){
    // id-muuttuja on asetettu osoiterivillä
}
```

Parametrin arvon tarkistus

- Parametrin arvon tarkistukseen käytetään normaalisti ehtolausetta. Huomaa, että useampia ehtoja voidaan niputtaa yhteen lauseeseen &&-operaattorin avulla:
- lauseke "if ($ehto1 && $ehto2)" on tosi ainoastaan mikäli $ehto1 on tosi JA $ehto2 on tosi.

Parametrin tarkistus - esimerkki

- Tarkistetaan, että parametri "luku1" on asetettu, ja että sen arvo on välillä 2...5:

```
if (isset($_GET['luku1'])){
    $luku = $_GET['luku1'];
    if ($luku >= 2 && $luku <= 5){
        echo 'Tarkastettu!';
    }
}
```

Parametrien käsittely samalla sivulla

- Usein on tarkoituksenmukaista käsitellä parametrit samalla sivulla kuin missä linkit ovat.
- Tällaisessa tapauksessa käsittelijä kirjoitetaan yleensä ennen sivun muodostavaa HTML-merkkausta.

Esimerkki:

```
<?php
      if (isset($_GET['linkki'])){
      $linkki = $_GET['linkki'];
      }
      else{
      $linkki = 'ei mitään';
      }
?>
<html>
      <body>
      <?php
      echo '<h1>Valittu linkki:' . $linkki . '</h1>';
      ?>
      <ul>
      <li><a href="oma.php?linkki=1">1. linkki</a></li>
      <li><a href="oma.php?linkki=2">2. linkki</a></li>
      <li><a href="oma.php?linkki=3">3. linkki</a></li>
      </ul>
      </body>
</html>
```

Lomakkeiden käsittely

- Komentoriviparametrien lisäksi tietoa voidaan välittää PHP:lle lomakkeiden avulla.
- Lomakkeiden välittämät arvot tallennetaan taulukkomuuttujaan $_POST.

Lomakkeen merkkaus

- Käsittelevän agentin (eli form-elementin action-parametrin) arvoksi tulee lomakkeen käsittelevän PHP-sivun osoite ja metodiksi "post":

```
<form method="post" action="handle.php"
id="henkilotiedot">
    <label for="nimi">Kirjoita nimesi:</label>
    <input name="nimi" type="text" size="30"
maxlength="40">
    <input name="laheta" type="submit" value="Lähetä!">
</form>
```

Lomakkeen käsittely (2)

- Edellisen lomakkeen lähettämä nimi saadaan $_POST –taulukosta name-attribuutin mukaisella avaimella:
- handle.php:

```
<?php
    echo 'Moi vaan, ' . $_POST['nimi'] . '!';
?>
```

Lomake-elementtien välittämät tiedot

- Eri lomake-elementit lähettävät tietoa käsittelevälle ohjelmalle seuraavasti:
  - Tekstikenttä (<input type="text">): Kentän sisältö tekstinä
  - Salasanakenttä (<input type="password">: Kentän sisältö tekstinä
  - Tekstialue (<textarea>): Kentän sisältö tekstinä
  - Asetusnappi (<input type="checkbox">): "on" jos nappi on valittu, muuten ei mitään
  - Optionapit (<input type="radio">) : Valitun option value-parametrin arvo
  - Lähetysnappi (<input type="submit">): value-kentän arvo, yleensä ei kuitenkaan lueta!

Lomakkeen käsittely (3)

- Huomaa, että samoin kuin komentoriviparametrien käsittelyssä, lomakkeen käsittely voidaan haluttaessa hoitaa samassa tiedostossa, kuin missä lomakkeen merkkauskin on.
- Samoin muuttujien olemassaolo (isset-funktiolla) ja arvot pitää aina tarkistaa ennen niiden käyttöä!