Iteration

Ibland behöver vi göra samma sak många gånger. Ibland behöver vi göra samma sak fast med en liten variation flera gånger. Så fort vi behöver göra någonting flera gånger bör vi se det som en ledtråd till att vi behöver **iterera**. Som tumregel kan du tänka att vi använder for-loopar när vi vet hur många varv vi vill "snurra", när vi har ett fast värde vi ska nå. `while` använder vi oftast när vi inte vet hur länge vi ska **iterera**.

`for`

```
for($i = 0; $i < 10; $i = $i + 1){
    echo $i;
}
```

När vi skapar en for-loop kan vi se det som att vi ger den tre saker separerade med semi-kolon (`;`). Dessa tre är i ovan exempel:

1. `$ = 0`;
2. `$i < 10`;
3. `$i++`;

Detta kan översätta ungefär till:

1. Skapa en variabel vi vill använda som räknare, en variabel som håller koll.
2. Definiera ett **condition**, hur länge denna loop ska pågå (tills räknaren har nått 10)
3. **Inkrementera** vår räknare, hur mycket vår räknare ska öka varje gång. I detta fall ökar vi med 1 varje gång så vår loop kommer att köras **10 gånger**.

Övningar

Loopar

1.

Använd loopen från innan, fast istället för att skriva ut varje siffra: Lägg ihop sifforna i en ny variabel samt skriv ut den variabeln med `echo` efter loopen är slut. Du ska alltså lägga ihop alla värden till en variabel `$sum`.

2.

Skapa en `for`-loop som skriver ut varannat tal. Loopen ska gå från 0 till 10. Använd loopen från ovan.

3.

Skriv en `for`-loop som skriver ut värden åt andra hållet, så att siffrorna skrivs ut **10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0**

4.

Skriv en `for`-loop med ett **condition**(`if`-sats) som gör så att endast siffror som är **jämna tal** skrivs ut till sidan.

5.

Skriv en `while`-loop som gör samma som övning 4.

6.

Vad är skillnaden på de här två scripten? Vad kommer de båda skriva ut och varför?

```
$num = 0;
while($num < 0){
    echo $num;
    $num++;
}
```

```
$num = 0;
do{
    echo $num;
    $num++;
}while($num < 0);
```

7.

Mina får förökar sig snabbt och jag behöver ett php-script som kan räkna ut hur många de kommer att vara inom ett år. Varje månad kommer fåren att multipliceras med 4.

Använd dessa tre variabler nedanför:

```
$numberOfSheep = 4;
$monthNumber = 1;
$monthsToPrint = 12;
```

För att sedan skriva ut detta för varje månad:

```
Output:
There will be 16 sheep after 1 month(s)!
There will be 64 sheep after 2 month(s)!
```

8.

Jag vill ha ett program som mjauar!

Programmet fungerar som så att den frågar användaren efter hur många mjau den vill ha. Om användaren skriver "3", ska programmet svara med "mjau mjau mjau". Om användaren skriver "4" ska programmet svara med "mjau mjau mjau mjau". Om användaren skriver "0" ska programmet "avslutas", d.v.s. inte ta in mer input. Programmet ska fungera som följande:

- Läs in ett tal från en variabel.
- Om talet är inte är `0` a. Skriv lika många *"mjau"* som talet, i rad
- Annars skriv ut "😾" eller något annat.

Arrays

Nedan kommer lite övningar för att öva er på att jobba med arrayer. För att skapa en array använder man följande syntax:

```
$myArray = array();
```

Detta skapar en tom array, `[]` indikerar att det är en array som skapas.

```
$myArray = array( 1, 2, 3);
```

Varje värde i en array är separerade med ett komma. Det ska dock inte vara ett komma efter det sista värdet. Detta är en array med tre värden. Värde 1 har index 0, värde 2 har index 1 samt värde 3 har index 2. Platsen i arrayen börjar räknas ifrån 0.

Övningar

1.

Vi ska börja med att skriva ut olika värden i en array. Om vi har en array som denna:

```
$your_array = array(23, 45, 54, 12, 77);
```

- Skriv ut det första och sista värdet (23 & 77) i denna array med hjälp av `echo`
- Vilka index ligger värdena på?

2.

Om vi redan har en array som den ovan kan vi även direkt ändra på ett visst värde på samma sätt som när vi tilldelar en variabel ett värde med `=`.

- Ändra sista värdet i `$your_array` genom att tilldela det nya värdet `1`.
- `echo`a arrayen efter att du har lagrat det nya värdet för att se att värdet verkligen har ändrats.

3.

För att komma åt alla värden i en array vill vi ju inte skriva in varenda index, speciellt inte om vi inte vet hur lång arrayen är, alltså hur många värden som finns inuti den. Vi kan inte bara skriva ut hela innehållet i arrayen med `echo $my_array` heller, det kommer bara att skriva ut hela arrayen och inte alla värden för sig. Tur att loopar finns.

Du har denna array:

```
$best_array = array(1, 2, 3, 4, 5);
```

Nu ska du loopa igenom arrayen och skriva ut varje värde i arrayen. Tänk på att längden av en array kan man ta ut med `count($best_array)` samt att varje värde i en array har ett index som man kommer åt värdet ifrån. Indexet är då detsamma som det nuvarande värdet på räknaren i loopen.

- **`count`** @ php.net

4.

Använd samma array som tidigare. Nu ska du dock loopa igenom arrayen och multiplicera varje värde i arrayen med summan av det föregående värdet. D.v.s, 1 * 2 * 3.. etc.

Spara värdet i en `$sum` och console.logga sedan ut denna variabel efter att loopen har körts klart

5.

Du ska utgå från följande array:

```
$ok_array = array("fine", "FINE", "good", "what is this stuff?", "sweet", "i don't even live here");
```

Du ska loopa igenom arrayen och console.logga dess värden. Dock ska din loop inte skriva ut strängar som är längre än 5 tecken. `"fine"`, `"FINE"`, `"good"` och `"sweet"` ska alltså skrivas ut men inte `"whatisthisstuff?"` och `"i don't even live here"`.

För att komma åt hur lång en sträng är kan man använda `strlen()`, en inbyggd funktion: **`strlen()`** @ php.net

6.

```
$worst_array_yet = array("string", true, 42, "another string", 54, true, 1);
```

För att få ut vilket värde en variabel är kan vi använda `is_string()` t.e.x. som returnerar true eller false baserat på om värdet är en sträng. Detta kan vi sedan använda i en if-sats.

Du ska loopa igenom arrayen `$worst_array_yet` och ska sedan `echoa` ut varje sträng som förekommer i arrayen. Men om värdet i arrayen är något annat ska det värdet läggas till i `$sum;` för att sedan efter att loopen är klar `echo`a ut.

Lösningsförslag

Loopar

1.

```
//them sum is zero from start
$sum = 0;
//i++ is a shorthand for writing '$i = $i + 1'
for($i = 0; $i <= 10; i++){
    //we must always add the new value + the old sum
    //otherwise we will only store the newest value
    $sum = $sum + $i;
}
```

2.

```
for($i = 0; $i < 10; $i = $i + 2){
    echo $i;
}
```

3.

```
//start from 10, and as long as $i is above 0, echo.
//instead of doing $i = $i + 1 we are doing $i = $i - 1; reverse!
for($i = 10; $i > 0; i--){
    echo $i;
}
```

4.

```
for($i = 0; $i < 10; $i++){
    if( $i % 2 == 0){
        echo $i;
    }
}
```

5.

```
$number = 0;

while($number < 10){
    //If it's a even number, we wont get rest value, we will have 0
    if( $number % 2 == 0){
        echo $number
    }
    //If we don't change the value of number, this loop will never end
    //the condition will always be met
    $number++;
}
```

6.

Eftersom vårt **condition** redan är mött så går vi aldrig in i `while`-loopen, vilket betyder att loopen aldrig körs. Men med en `do while` så går vi alltid in i loopen minst 1 gång, SEDAN kollar vi villkoret. Villkoret stämmer fortfarande inte men då har vi redan hunnit köra vår `do`.

7.

```
$numberOfSheep = 4;
$monthNumber = 1;
$monthsToPrint = 12;

for ($monthNumber; $monthNumber <= $monthsToPrint; $monthNumber++){

    $numberOfSheep = $numberOfSheep * 4;
    echo 'There will be ' . $numberOfSheep . ' sheeps after ' . $monthNumber +' months(s)!';
}
```

8.

```
$number_of_mjau = 10;
if($number_of_mjau == 0){
    echo '😿';
} else {
    $all_the_mjaus = '';
    for($i = 0; $i <= $number_of_mjau; $i++){
        $all_the_mjaus = $all_the_mjaus . 'mjau ';
    }
    echo $all_the_mjaus;
}
```

Arrayer

1.

```
$your_array = array(23, 45, 54, 12, 77);
echo $your_array[0]; //index 0
echo $your_array[4]; //Last index is 4, but length is 5
```

2.

```
$your_array = array(23, 45, 54, 12, 77);
$your_array[0] = 1;
echo $your_array[4];
```

3.

```
$best_array = array(1, 2, 3, 4, 5);
//The number of times the loop will runt is based on the length of the array
//5 items in the array == count($best_array) returns 5.
for($i = 0; $i < count($best_array); $i++){
    //$i is 0,1,2,3,4, this can be used to access the value at these indexes
    echo $best_array[$i];
}
```

4.

```
$best_array = array(1, 2, 3, 4, 5);
$sum = 0;
//The number of times the loop will runt is based on the length of the array
//5 items in the array == count($best_array) returns 5.
for($i = 0; $i < count($best_array); $i++){
    //$i is 0,1,2,3,4, this can be used to access the value at these indexes
    $sum = $sum + $best_array[$i];
}
echo $sum;
```

5.

```
$ok_array = array("fine", "FINE", "good", "what is this stuff?", "sweet", "i don't even live here");

for($i = 0; $i < count($ok_array); $i++){
    //$i is 0,1,2,3,4, this can be used to access the value at these indexes
    $current_string = $ok_array[$i];
    if(strlen($current_string) <= 5){
        echo $current_string;
    }
}
```

6.

```
$worst_array_yet = array("string", true, 42, "another string", 54, true, 12);
$sum = 0;

for($i = 0; $i < count($worst_array_yet); $i++){
    //$i is 0,1,2,3,4, this can be used to access the value at these indexes
    $current_value = $ok_array[$i];
    if(is_string($current_value)){
        echo $current_value;
    } else{
        $sum = $sum + $current_value;
    }
}

echo $sum;
```