מחרוזות תווים

יצירת מחרוזת תווים

מחרוזת תווים ב-PHP יכולה לשמש, פרט לאמצעי להחזקתו של טקסט, גם כאמצעי להחזקתו של מידע מכל סוג אחר. קיימות מספר דרכים ליצירת מחרוזת תווים ב-PHP.

הדרך הפשוטה יותר כוללת שימוש בגרשיים כפולים (או גרשיים בודדים) באופן הבא:

```
$str = 'abc';
$str = "abc";
```

כאשר משתמשים בגרשיים כפולים אז ניתן לשלב בתוך המחרוזת שיוצרים את ערכו של משתנה.

```
$str = "number=$number";
```

כאשר משתמשים בגרשיים כפולים אז ניתן לשלב בתוך המחרוזת characters מיוחדים כגון '\n' ואחרים.

```
$str = "Hello\nWorld";
```

את שם המשתנה ניתן גם למקם בתוך סוגריים מסולסלות. לעתים זו תהיה הדרך להתמודד עם מקרים בהם מנוע ה-PHP מתקשה לשלב את ערכו של המשתנה בתוך מחרוזת התווים.

```
$str = "number={$number}";
```

כאשר יוצרים מחרוזת תווים באמצעות גרשיים בודדים ורוצים לשלב בתוכה גרשיים בודדים ניתן לעשות זאת באמצעות backslash.

```
$str = 'we can create new strings using \' having it on both sides';
```

כאשר יוצרים מחרוזת תווים באמצעות גרשיים כפולים אז ניתן לשלב בתוכה גרשיים בודדים ללא כל בעיה.

```
$str = "we can create new strings using ' having it on both sides";
```

כאשר יוצרים מחרוזת תווים באמצעות גרשיים כפולים ורוצים לשלב בתוכה את הסימן $ ו/או גרשיים כפולים ניתן לעשות זאת באמצעות הוספת backslash.

```
$str = "we can include \$ sigh as well as \" within our string";
```

קיימת דרך נוספת ליצירת מחרוזות תווים בשם Heredoc. בדרך זו אנו משתמשים באופרטור <<< ומייד אחריו רושמים identifier ולאחריו רושמים את הטקסט אשר יכול להשתרע על פני מספר שורות. בסוף הטקסט (חייב להופיע בתחילת שורה) אנו רושמים שוב את אותו identifier ומייד אחריו ;.

הדוגמא הבאה מציגה את אופן יצירתו של string באמצעות Heredoc.

```
<?php

$text_var = <<< TOKTOK

We wish you a happy birthday!

All the best!

Regards,

Friends.

TOKTOK;

echo $text_var;

?>
```

הפלט שנקבל הוא:

```
We wish you a happy birthday! All the best! Regards, Friends.
```

הפונקציה strlen

באמצעות פונקציה זו ניתן לקבל את האורך של מחרוזת תווים (בבתים).

```
<?php
$text_1 = "abcdef";
$text_2 = "abc def ";
$text_3 = "\n";
$length_1 = strlen($text_1);
$length_2 = strlen($text_2);
$length_3 = strlen($text_3);
echo "<BR>The length of \$text_1 is $length_1";
echo "<BR>The length of \$text_2 is $length_2";
echo "<BR>The length of \$text_3 is $length_3";
?>
```

הפלט שנקבל הוא:

```
The length of $text_1 is 6
The length of $text_2 is 8
The length of $text_3 is 1
```

הפונקציה strtr

באמצעות פונקציה זו ניתן להחליף כל אחד מהתווים שכלולים בתת מחרוזת מסויימת בתווים המתאימים שכלולים בתת המחרוזת האחרת ובדרך זו לקבל מחרוזת תווים חדשה.

לפונקציה זו קיימות שתי גרסאות.

גרסה אחת כוללת שלושה פרמטרים. הפרמטר הראשון הוא המחרוזת שנתונה לנו. הפרמטר השני הוא תת המחרוזת שכל תו שמופיע בה יוחלף בתו המתאים מתת המחרוזת האחרת (הפרמטר השלישי). התכנית הבאה מדגימה את אופן השימוש בגרסה זו של הפונקציה.

```
<?php
$text_1 = "shalom is a nice guy that likes saying shalom";
$text_2 = strtr($text_1,"shalom","sholosholo");
echo $text_2;
?>
```

הפלט שנקבל הוא:

```
sholos is o nice guy thot likes soying sholos
```

במקרה שתת המחרוזת השניה ארוכה מהראשונה אז ההחלפה תתעלם מכל התווים העודפים.

גרסה שניה כוללת שני פרמטרים. הפרמטר הראשון הוא המחרוזת הנתוניה. הפרמטר השני הוא מערך שבו כל `key` יוחלף ב-`value` שלו. הדוגמא הבאה מציגה את אופן השימוש בגירסה זו.

```
<?php
$text_1 = "shalom is a nice guy that likes saying shalom";
$text_2 = strtr($text_1,array("sha"=>"cho","is"=>"as"));
echo $text_1;
echo "<br>";
echo $text_2;
?>
```

הפלט שנקבל הוא:

```
shalom is a nice guy that likes saying shalom
cholom as a nice guy that likes saying cholom
```

מחרוזת תווים כמערך

ניתן להתייחס לכל מחרוזת תווים כאל מערך. הדוגמא הבאה מציגה זאת.

```
<?php
	$str = "abcdefghijklmnopqrstuvwxyz";
	$length = strlen($str);
	for($index=0; $index<$length; $index++)
		echo "<BR>$str[$index]";
?>
```

הפלט שיתקבל הוא:

a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p
q
r
s
t
u
v
w
x
y
z

השוואה בין מחרוזות תווים

השוואה של שתי מחרוזות תווים באמצעות האופרטור == כרוכה בתרגום של כל אחת משתי המחרוזות לערך מספרי והשוואת שני הערכים המספריים שמתקבלים. מסיבה זו, שימוש באופרטור == לצורך השוואה של מחרוזת תווים נתונה למשתנה שמחזיק בערך מספרי שזהה לערך שאליו מחרוזת התווים תומר תחזיר true. הדוגמא הבאה מציגה זאת.

```
<?php
$str_1 = '1978yearsdfds';
$str_2 = 1978;
echo "\$str_1=$str_1";
echo "<br>";
echo "\$str_2=$str_2";
echo "<br>";
if($str_1==$str_2)
{
      echo "'$str_1' and '$str_2' are equal";
}
else
{
      echo "'$str_1' and '$str_2' are not equal";
}
?>
```

הפלט שנקבל הוא:

```
$str_1=1978yearsdfds
$str_2=1978
'1978yearsdfds' and '1978' are equal
```

השוואה בין מחרוזות תווים באמצעות === איננה כרוכה בהמרה אוטומאטית של המחרוזות לטיפוס נומרי (או אחר). הדוגמא הבאה מציגה זאת.

```
<?php
$str_1 = '1978xyz';
$str_2 = 1978;
$str_3 = "1978xyz";
echo "\$str_1=$str_1";
echo "<br>";
echo "\$str_2=$str_2";
echo "<br>";
echo "\$str_3=$str_3";
echo "<br>";
if($str_1===$str_2)
{
      echo "<br>'\$str_1' and '\$str_2' are equal";
}
else
{
      echo "<br>'\$str_1' and '\$str_2' are not equal";
}
if($str_1===$str_3)
{
      echo "<br>'\$str_1' and '\$str_3' are equal";
}
else
{
      echo "<br>'\$str_1' and '\$str_3' are not equal";
}
?>
```

הפלט שנקבל הוא:

```
$str_1=1978xyz
$str_2=1978
$str_3=1978xyz

'$str_1' and '$str_2' are not equal
'$str_1' and '$str_3' are equal
```

ניתן להשוות בין שתי מחרוזות תווים באמצעות הפונקציות strcmp ו-strcasecmp. לכל אחת מהן שני פרמטרים. שתיהן מקבלות שתי מחרוזות תווים ומבצעות השוואה ביניהן. אם שתי מחרוזות התווים זהות אז מוחזר הערך 0. אם לא, אז מוחזר מספרי חיובי או ערך מספרי שלילי בהתאם ליחס הלקסיקוגרפי ביניהן. ההבדל בין שתי הפונקציות הללו הוא שהראשונה היא case sensitive והשניה לא. הדוגמה הבאה מציגה את אופן השימוש בשתי פונקציות אלה.

```
<?php
$str_1 = "moshe";
$str_2 = "haim";
$str_3 = "Haim";
echo "\$str_1=$str_1";
echo "<br>";
echo "\$str_2=$str_2";
echo "<br>";
echo "\$str_3=$str_3";
echo "<br>";
echo "<br>strcmp(\$str_1,\$str_2)=".strcmp($str_1,$str_2);
echo "<br>strcmp(\$str_2,\$str_1)=".strcmp($str_2,$str_1);
echo "<br>strcmp(\$str_1,\$str_3)=".strcmp($str_1,$str_3);
echo "<br>strcmp(\$str_3,\$str_1)=".strcmp($str_3,$str_1);
echo "<br>strcmp(\$str_2,\$str_3)=".strcmp($str_2,$str_3);
echo "<br>strcmp(\$str_3,\$str_2)=".strcmp($str_3,$str_2);
echo "<br>strcasecmp(\$str_1,\$str_2)=".strcasecmp($str_1,$str_2);
echo "<br>strcasecmp(\$str_2,\$str_1)=".strcasecmp($str_2,$str_1);
echo "<br>strcasecmp(\$str_1,\$str_3)=".strcasecmp($str_1,$str_3);
echo "<br>strcasecmp(\$str_3,\$str_1)=".strcasecmp($str_3,$str_1);
echo "<br>strcasecmp(\$str_3,\$str_2)=".strcasecmp($str_3,$str_2);
echo "<br>strcasecmp(\$str_2,\$str_3)=".strcasecmp($str_2,$str_3);
?>
```

הפלט שמקבלים הוא:

```
$str_1=moshe
$str_2=haim
$str_3=Haim

strcmp($str_1,$str_2)=1
strcmp($str_2,$str_1)=-1
strcmp($str_1,$str_3)=1
strcmp($str_3,$str_1)=-1
```

```
strcmp($str_2,$str_3)=1
strcmp($str_3,$str_2)=-1
strcasecmp($str_1,$str_2)=5
strcasecmp($str_2,$str_1)=-5
strcasecmp($str_1,$str_3)=5
strcasecmp($str_3,$str_1)=-5
strcasecmp($str_3,$str_2)=0
strcasecmp($str_2,$str_3)=0
```

הפונקציה `strncmp` דומה בפעולתה לפונקציה `strcmp` פרט להבדל אחד. אנו יכולים לציין את מספר התווים שיילקחו בחשבון בעת ביצוע ההשוואה. הפונקציה `strncasecmp` דומה בפעולתה פרט לכך שהיא איננה `case sensitive`. הדוגמא הבאה מדגימה את אופן השימוש בפונקציה `strncmp`.

```
<?php
$str_1 = "moshe";
$str_2 = "moshico";
echo "\$str_1=$str_1";
echo "<br>";
echo "\$str_2=$str_2";
echo "<br>";
echo "<br>strncmp(\$str_1,\$str_2,3)=".strncmp($str_1,$str_2,3);
?>
```

הפלט שנקבל הוא:

```
$str_1=moshe
$str_2=moshico
strncmp($str_1,$str_2,3)=0
```

חיפוש בתוך מחרוזת תווים

באמצעות הפונקציה strpos ניתן לקבל את המיקום שבו תת מחרוזת מופיעה בתוך מחרוזת נתונה. הארגומנט הראשון ששולחים הוא המחרוזת הנתונה שבתוכה רוצים לבצע את החיפוש. הארגומנט השני ששולחים הוא תת המחרוזת שרוצים לחפש בתוך המחרוזת הנתונה. ניתן לשלוח ארגומנט שלישי שהוא המיקום שהחל ממנו יחל החיפוש. הדוגמא הבאה מציגה שימוש פשוט בפונקציה זו.

```
<?php
$str_1 = "972 54 6655837 #243";
$str_2 = "#";
echo "\$str_1=$str_1";
echo "<br>";
echo "\$str_2=$str_2";
echo "<br>";
echo "<br>strpos(\$str_1,\$str_2)=".strpos($str_1,$str_2);
?>
```

הפלט שנקבל מהפעלת הפונקציה הוא:

```
$str_1=972 54 6655837 #243
$str_2=#
strpos($str_1,$str_2)=15
```

הפונקציה `strstr` מקבלת שני ארגומנטים. הארגומנט הראשון הוא המחרוזת הנתונה שבה רוצים לבצע את החיפוש. הארגומנט השני הוא תת המחרוזת שרוצים לחפש בתוך המחרוזת הנתונה. הפונקציה מחזירה את תת המחרוזת שמתחילה במספר האינדקס בו נמצאת ההתאמה ועד לסוף המחרוזת הנתונה שבה מתבצע החיפוש. ניתן לשלוח ארגומנט נוסף (ארגומנט שלישי) מטיפוס `boolean` ולהורות באמצעותו (במידה ששולחים `true`) כי ברצוננו לקבל בחזרה תת מחרוזת שנגזרת מהמחרוזת הנתונה מתחילתה ועד למספר האינדקס בו נמצאת ההתאמה. הדוגמא הבאה מציגה שימוש פשוט בפונקציה זו.

```
<?php
$str_1 = "972 54 6655837 #243";
$str_2 = "#";
echo "\$str_1=$str_1";
echo "<br>";
echo "\$str_2=$str_2";
echo "<br>";
echo "<br>strstr(\$str_1,\$str_2)=".strstr($str_1,$str_2);
?>
```

הפלט שמתקבל הוא:

```
$str_1=972 54 6655837 #243
$str_2=#
strstr($str_1,$str_2)=#243
```

הפונקציות stripos ו- stristr מבצעות את אותה עבודה שמבצעות הפונקציות strpos ו-strstr בהבדל אחד. הן לא רגישות לאותיות גדולות

הפונקציה strrpos עושה את אותה פעולה שהמתודה strpos מבצעת בהבדל אחד. המתודה strrpos מתחילה את החיפוש בסדר הפוך (מהסוף להתחלה).

הפונקציה strspn מחזירה את האורך של תת המחרוזת (מתוך המחרוזת הנתונה שנשלחת אליה כארגומנט ראשון) אשר מורכבת מתווים שמופיעים בתוך תת המחרוזת שנשלחת בתור הארגומנט השני אל הפונקציה.

הדוגמא הבאה מציגה שימוש בסיסי בפונקציה זו.

```
<?php
echo strspn("AAwaXfwaxAfaaaaoo", "fwaAo");
?>
```

הפלט שנקבל הוא:

```
4
```

פלט זה מתייחס לתת המחרוזת AAwa

הפונקציה strcspn מבצעת את הפעולה ההפוכה ומחזירה את האורך של תת המחרוזת (מתוך המחרוזת הנתונה) שאיננה כוללת אף אחד מהתווים שכלולים בתת המחרוזת שנשלחת כארגומנט שני.

הדוגמא הבאה מציגה שימוש בסיסי בפונקציה זו.

```
<?php
echo strcspn("ddsAAwavbnmfwaxAfaaaaoo", "fwaAo");
?>
```

הפלט שנקבל הוא:

```
3
```

פלט זה מתייחס לתת המחרוזת dds.

שינוי מחרוזת תווים

באמצעות הפונקציה str_replace ניתן להחליף את כל אחד מהמופעים של כל אחת מתת המחרוזות שאנו מחפשים בתת מחרוזות אחרות. הארגומנט הראשון שנשלח הוא מערך של תת מחרוזות שאנו רוצים לחפש במחרוזת הנתונה. הארגומנט השני הוא מערך של תת מחרוזות שאליהן אנו רוצים בהתאמה להחליף כל מופע של כל אחת מתת המחרוזות שמופיעות במערך ששלחנו כארגומנט ראשון. הארגומנט השלישי הוא מחרוזת התווים הנתונה שבה אנו רוצים לבצע את ההחלפות.

התכנית הבאה מדגימה את אופן השימוש הבסיסי בפונקציה זו.

```
<?php
	$str = "shalom abba shalom eema";
	$vec_search = array("shalom","abba","eema");
	$vec_replace = array("hello","father","mother");
	$new_str = str_replace($vec_search,$vec_replace,$str);
	echo $new_str;
?>
```

הפלט שנקבל:

```
hello father hello mother
```

ניתן לשלוח כארגומנט רביעי משתנה (by reference) אשר יתמלא בערך מספרי שהוא מספר ההחלפות שבוצעו. הדוגמא הבאה מציגה את אופן השימוש בו.

```
<?php
	$num = 0;
	$str = "shalom abba shalom eema";
	$vec_search = array("shalom","abba","eema");
	$vec_replace = array("hello","father","mother");
	$new_str = str_replace($vec_search,$vec_replace,$str,$num);
	echo "\$new_str=$new_str";
	echo "<br>";
	echo "\$num=$num";
?>
```

הפלט שנקבל:

```
$new_str=hello father hello mother
$num=4
```

הפונקציה substr_replace מאפשרת לנו להחליף תת מחרוזת מסויימת מתוך מחרוזת נתונה בתת מחרוזת אחרת. הארגומנט הראשון שיש לשלוח אל הפונקציה הוא המחרוזת הנתונה שבה אנו רוצים לבצע את השינויים. בפועל, הפונקציה לא תשנה אותה... היא רק תחזיר מחרוזת חדשה שתתקבל כתוצאה מהחלפת תת מחרוזת מסויימת באחרת. הארגומנט השני שיש לשלוח הוא תת המחרוזת שיש לשלב במחרוזת הנתונה (הארגומנט הראשון). הארגומנט השלישי הוא המיקום במחרוזת הנתונה (הארגומנט הראשון) שהחל ממנו יש לשלב את תת המחרוזת (הארגומנט השני). הארגומנט הרביעי (אופציונלי) הוא מספר התווים מתוך המחרוזת הנתונה (הארגומנט הראשון) שיוחלפו בתת המחרוזת (הארגומנט השני). אם לא שולחים את הארגומנט הרביעי אז כל המחרוזת הנתונה (הארגומנט הראשון) החל מהמיקום שיצויין (הארגומנט השלישי) תוחלף בתת המחרוזת (הארגומנט השני).

התכנית הבאה מדגימה את אופן השימוש הבסיסי בפונקציה זו.

```
<?php
$var = 'abcdefghijklmnopqrstuvwxyz';
echo "original string...<BR>$var<BR>";
echo "<BR><BR>substr_replace(\$var, 'MOSHE', 0)<BR>";
echo substr_replace($var, 'MOSHE', 0);
echo "<BR><BR>substr_replace(\$var, 'MOSHE', 10)<BR>";
echo substr_replace($var, 'MOSHE', 10);
echo "<BR><BR>substr_replace(\$var, 'MOSHE', 10, 2)<BR>";
echo substr_replace($var, 'MOSHE', 10, 2);
echo "<BR><BR>substr_replace(\$var, 'MOSHE', 0, 0)<BR>";
echo substr_replace($var, 'MOSHE', 0, 0);
echo "<BR><BR>substr_replace(\$var, 'MOSHE', 5, 1)<BR>";
echo substr_replace($var, 'MOSHE', 5, 1);
?>
```

הפלט שיתקבל:

```
original string...
abcdefghijklmnopqrstuvwxyz
substr_replace($var, 'MOSHE', 0)
MOSHE
substr_replace($var, 'MOSHE', 10)
abcdefghijMOSHE
substr_replace($var, 'MOSHE', 10, 2)
abcdefghijMOSHEmnopqrstuvwxyz
substr_replace($var, 'MOSHE', 0, 0)
MOSHEabcdefghijklmnopqrstuvwxyz
substr_replace($var, 'MOSHE', 5, 1)
abcdeMOSHEghijklmnopqrstuvwxyz
```

קבלת תת מחרוזת

הפונקציה `substr` מחלצת תת מחרוזת מתוך מחרוזת נתונה בהתאם לארגומנטים שנשלחים אליה. הארגומנט הראשון הוא המחרוזת שמתוכה ברצוננו לחלץ תת מחרוזת. הארגומנט השני הוא המיקום במחרוזת הנתונה (הארגומנט הראשון) שהחל ממנו אנו רוצים לחלץ תת מחרוזת. הארגומנט השלישי הוא האורך של תת המחרוזת שאנו רוצים לקבל.

הדוגמא הבאה מציגה אופן שימוש בסיסי בפונקציה זו:

```
<?php
$var = 'abcdefghijklmnopqrstuvwxyz';
echo "original string<BR>$var";
echo "<BR><BR>substr(\$var,0)<BR>";
echo substr($var,0);
echo "<BR><BR>substr(\$var,0,4)<BR>";
echo substr($var,0,4);
?>
```

הפלט שנקבל:

```
original string
abcdefghijklmnopqrstuvwxyz
substr($var,0)
abcdefghijklmnopqrstuvwxyz
substr($var,0,4)
abcd
```

לוקליזציה

האופן שבו יש להציג את התכנים (תאריכים, מספרים וכו') עשוי להשתנות בהתאם למיקום הגיאוגרפי. באמצעות הפונקציה setLocale ניתן לקבוע את המיקום הגיאורפי שאליו הקוד צריך להתייחס ובדרך זו להשפיע למעשה על אופן הפעולה של הפונקציות השונות. הארגומנט השני שיש לשלוח לפונקציה זו הוא שם ה-locale (או מערך של שמות locale כדי שיהיה ניסיון לקבוע אחד מהם.. על פי הסדר). הארגומנט הראשון הוא ערך מספרי שלם אשר מייצג את הקטגוריה (או הקטגוריות) של הפונקציות אשר תושפענה מהפעלת הפונקציה setLocale.

הארגומנט הראשון הוא ערך מספרי שלם שמתקבל מביצוע פעולת OR ברמה של ביטים על הערכים הקבועים השלמים שמייצגים את הקטגוריות השונות האפשריות:

LC_ALL for all of the below
LC_COLLATE for string comparison, see strcoll()
LC_CTYPE for character classification and conversion, for example strtoupper()
LC_MONETARY for localeconv()
LC_NUMERIC for decimal separator (See also localeconv())
LC_TIME for date and time formatting with strftime()
LC_MESSAGES for system responses (available if PHP was compiled with *libintl*)

הארגומנט השני יכול להיות צירוף של שתי אותיות אשר מייצגות שפה (en לדוגמא) בצירוף קו תחתי ושתי (או שלוש) אותיות שמייצגות מדינה (US לדוגמא). צירופי האותיות האפשריים מפורטים ב-ISO 639.

את ISO 639 ניתן למצוא בכתובת ה-URL הבאה:

http://www.w3.org/WAI/ER/IG/ert/iso639.htm

להלן דוגמא אפשרית לאופן הפעלת הפונקציה האמורה.

```
setLocale(LC_ALL,'en_US');
```

שליטה באופן התצוגה של מספרים נעשית באמצעות הפעלת הפונקציה number_format. פונקציה זו ניתנת להפעלה תוך שליחה של ארגומנט אחד, שניים או ארבעה. במקרה ששולחים ארבעה ארגומנטים, הארגומנט הראשון הוא הערך המספרי שרוצים להציג. הארגומנט השני (decimals) הוא מספר הספרות אחרי הנקודה. הארגומנט השלישי הוא הסימן המפריד שישמש להפרדה בין הערך השלם והערך העשרוני. הארגומנט הרביעי הוא הסימן המפריד בין האלפים.

הדוגמא הבאה מציגה זאת:

```
<?php
$num = 22423224.12344322;
$de_DE_num = number_format($num,4,",",".");
echo "<br>\$de_DE_num=$de_DE_num";
?>
```

הפלט שנקבל הוא:

```
$de_DE_num=22.423.224,1234
```

שליטה באופן התצוגה של ערכים כספיים נעשית באמצעות הפעלת הפונקציה money_format. פונקציה זו ניתנת להפעלה תוך שליחה של שני ארגומנטים. הארגומנט הראשון הוא הפורמט שבו אנו רוצים להשתמש. הארגומנט השני הוא הערך המספרי שאנו רוצים להציג. התכנית הבאה מציגה את אופן השימוש בפונקציה זו.

```
<?php
      $number = 125455.3244442;
      echo "<BR>en_US<BR>";
      setlocale(LC_MONETARY, 'en_US');
      echo money_format('%(#10n', $number) . "<BR>";
      echo "<BR>it_IT<BR>";
      setlocale(LC_MONETARY, 'it_IT');
      echo money_format('%i', $number) . "<BR>";
      echo "<BR>de_DE<BR>";
      setlocale(LC_MONETARY, 'de_DE');
      echo money_format('%i', $number) . "<BR>";
?>
```

הפלט שנקבל הוא:

```
en_US
$ 125,455.32
it_IT
EUR 125.455,32
de_DE
125.455,32 EUR
```

פונקציות printf

משפחת פונקציות זו כוללת פונקציות אשר מאפשרות לנו להציג טקסטים בפורמטים שונים. משפחה זו כוללת שלוש משפחות של פונקציות.

פונקציות printf אשר כותבות את הטקסט המתקבל ל-output הרגיל (חזרה לדפדפן).

פונקציות sprintf אשר מחזירות את הטקסט המתקבל.

פונקציות fprintf אשר כותבות את הטקסט המתקבל לקבצים.

באופן בסיסי, פונקציות אלה מאפשרות לשלוח כארגומנט מחרוזת תווים (format) אשר מורה באיזה פורמט להציג את הטקסט.
מחרוזת תווים זו יכולה לכלול מגוון של סימנים אשר ישפיעו על הפורמט.

תווים רגילים – ordinary characters
כל תו רגיל שנכלול בפורמט שאנו שולחים לפונקציה יופיע בתוצאה המתקבלת.

תווי המרה – conversion specification
כל תו המרה מורכב מהסימן % שבהמשכו אחד (או יותר) מהאלמנטים הבאים:
תו הסימן - sign specifier
תו ריפוד – padding specifier
תו יישור – padding specifier
תו רוחב – width specifier
תו דיוק – precision specifier
תו סוג – type specifier

תו הסימן
תו זה יכול להיות אחד מהשניים: '-' או '+'.
הדוגמא הבאה מציגה שימוש באפשרות זו:

```
<?php
$num = "12200";
printf("%+d",$num);
?>
```

הפלט שמתקבל הוא:

```
+12200
```

תו הריפוד - padding specifier

תו זה יכול להיות אחד מהשניים: ' ' או '0'. תו זה יופיע תוך שהוא חוזר על עצמו בריווח שמתקבל בטקסט המוצג. אפשר להשתמש בתו אחר ובתנאי שנוסיף ' לפניו.

הדוגמא הבאה מציגה שימוש בסיסי באפשרות זו:

```
<?php
	$year = 2009;
	$month = 6;
	$day = 9;
	printf("%04d-%02d-%02d", $year, $month, $day);
?>
```

הפלט שנקבל הוא:

```
2009-06-09
```

תו היישור - alignment specifier

באמצעות הוספת התו '-' נקבל יישור לשמאל. ברירת המחדל היא יישור לימין.

הדוגמא הבאה מציגה שימוש בסיסי באפשרות זו.

```
<?php
$str = 'momo';
printf("<BR>[%s]\n",$str);
printf("<BR>[%10s]\n",$str);
printf("<BR>[%-10s]\n",$str);
printf("<BR>[%010s]\n",$str);
printf("<BR>[%'#10s]\n",$str);
?>
```

הפלט שנקבל הוא:

[momo]
[momo]
[momo]
[000000momo]
[######momo]

תו רוחב - width specifier

באמצעות ציון מספר שלם ניתן להגדיר את מספר התווים שהטקסט המתקבל יתפוס (לפחות). הדוגמא הקודמת מדגימה זאת.

תו דיוק - precision specifier

כאשר מדובר בהצגת מספר ממשי ניתן באמצעות '.' לציין את מספר הספרות שתוצגנה לאחר הנקודה. הדוגמא הבאה מציגה אפשרות זו.

```
<?php
      $num = 129.3432;
      printf("<BR>[%.2f]\n",$num);
?>
```

הפלט שמתקבל:

```
[129.34]
```

תו סוג - type specifier

באמצעות ציון ה-type נוכל לגרום להצגת הערך בהתאם. האפשרויות הקיימות הן:

ג – מספר שלם

c – תו

d – מספר שלם רגיל

e – מספר בפורמט מדעי, לדוגמא: 1.5e44

u – מספר שלם חיובי

f – מספר ממשי שמוצג בהתאם ל-locale

F – מספר ממשי שמוצג ללא כל קשר ל-locale

o – מספר שלם שמוצג בבסיס אוקטלי

s – מחרוזת תווים

x – מספר שלם שמוצג בבסיס 16 תוך שימוש באותיות קטנות

X – מספר שלם שמוצג בבסיס 16 תוך שימוש באותיות גדולות

הדוגמאות עד כה הציגו שימוש באפשרויות אלה.

להלן דוגמא מפורטת לשימוש ב אפשרויות הפורמט השונות:

```
<?php
$positive_number = 43951789;
$negative_number = -43951789;
$simple_character = 65; // ASCII 65 is 'A'
$simple_string = "mosh";
$long_string = "moshiko goes to seven grade";
printf("%%b = '%b'<BR>", $positive_number);
printf("%%c = '%c'<BR>", $simple_character);
printf("%%d = '%d'<BR>", $positive_number);
printf("%%e = '%e'<BR>", $positive_number);
printf("%%u = '%u'<BR>", $positive_number);
printf("%%u = '%u'<BR>", $negative_number);
printf("%%f = '%f'<BR>", $positive_number);
printf("%%o = '%o'<BR>", $positive_number);
printf("%%s = '%s'<BR>", $positive_number);
printf("%%x = '%x'<BR>", $positive_number);
printf("%%X = '%X'<BR>", $positive_number);
printf("_%s_<BR>", $simple_string);
printf("_%20s_<BR>", $simple_string);
printf("_%-20s_<BR>", $simple_string);
printf("_%020s_<BR>", $simple_string);
printf("_%'#20s_<BR>", $simple_string);
echo sprintf("%.3e<BR>", $positive_number);
?>
```

הפלט שנקבל הוא:

```
%b = '10100111101010011010101101'
%c = 'A'
%d = '43951789'
%e = '4.395179e+7'
%u = '43951789'
%u = '4251015507'
%f = '43951789.000000'
%o = '247523255'
%s = '43951789'
%x = '29ea6ad'
%X = '29EA6AD'
_mosh_
_ mosh_
_mosh _
_0000000000000000mosh_
_################mosh_
4.395e+7
```

הפונקציה sscanf

פונקציה זו יודעת לקבל מחרוזת תווים בהתאם לפורמט שאנו קובעים. אופן קביעת הפורמט דומה לאופן קביעת הפורמט כאשר עובדים עם printf. מחרוזת התווים שנשלחת חייבת לתאום לפורמט שהוגדר.

להלן דוגמא שמציגה אופן שימוש בסיסי בפונקציה זו.

```
<?php
      $person = "38\t\tMichael Abelski";
      $ver = sscanf($person, "%d\t\t%s %s", $age, $first, $last);
      echo "$age,$first,$last";
?>
```

הפלט שמקבלים:

```
38,Michael,Abelski
```