1) Uruchamianie skryptu php
    a) http://localhost/nazwapliku.php
    b) http://127.0.0.1/nazwapliku.php

2) Komentarze
    a) blokowy: /* wielowierszowy */
    b) jednowierszowy //komentarz
    c) jednowierszowy uniksowy #komentarz

3) STRING – typ łańcuchowy, służy do zapamiętywania sekwencji znaków (np. tekst). Łańcuchy znaków możemy tworzyć na 4 sposoby, za pomocą:

Apostrofu

Cudzysłowu

Składni heredoc

Składni nowdoc

Po zastosowaniu składni heredoc łańcuch znakowy rozpoczyna się od sekwencji <<<, po której występuje identyfikator. Tworzony jest na tych samych zasadach co zmienne. Tego samego identyfikatora należy użyć na końcu łańcucha znakowego. Linia kończąco nie może zawierać nic poza identyfikatorem i znacznikiem.

Przykład

```php
<?php
$a=5;
$b=7;
$pole=$a*$b;
$heredoc = <<<ABC
wymiary prostokąta $a, $b <br>
a pole wynosi $pole;
ABC;
echo $heredoc;

$nowdoc = <<<'AB'
nazwa identyfikatora w apostrofach - nazwy zmiennych wypisane-brak podmiany na
wartości
wymiary prostokąta $a, $b <br>
a pole wynosi $pole;
AB;
echo $nowdoc;
?>
```

wymiary prostokąta 5, 7
a pole wynosi 35;wymiary prostokąta $a, $b
a pole wynosi $pole;

4) Funkcje formatowania ciągów

Pracując z ciągami znakowymi, często trzeba formatować je tak, aby były wyświetlane w określony sposób. Do tego celu można wykorzystywać funkcje formatujące. W języku PHP istnieje wiele funkcji, za pomocą których możemy ustalić określony wygląd ciągu.

Funkcja nl2br()

Jeżeli wyświetlamy w przeglądarce blok tekstu, który zawiera znaki końca linii, to przeglądarka nie uwzględni tych znaków. Można ominąć ten problem, dodając znaczniki ‹br› lub ‹p›. Ale nie zawsze takie rozwiązanie jest możliwe, szczególnie wtedy, gdy tekst jest wczytywany z pliku lub bazy danych. W takiej sytuacji można wykorzystać funkcję nl2br()

```
27    $nowy="assddf
28    ddfggh
29    gghjj   fgdfgndf dfgnd
30    fgdfg
31    dfgd
32    ------ostatnia linia
33    ";
34
35    echo $nowy;
36    echo "<br>";
37    echo nl2br($nowy);
```

Wynik linii 35

assddf ddfggh gghjj fgdfgndf dfgnd fgdfg dfgd ------ostatnia linia

Wynik linii 37 -strona internetowa

assddf
ddfggh
gghjj fgdfgndf dfgnd
fgdfg
dfgd
------ostatnia linia

```
<br>
"assddf"
<br>
" ddfggh"
<br>
" gghjj fgdfgndf dfgnd"
<br>
" fgdfg"
<br>
" dfgd"
<br>
" ------ostatnia linia"
<br>
```

działanie funkcji nl2br: dodae znaczniki

Funkcja wordwrap()

Funkcja wordwrap() służy do formatowania tekstu w postaci kolumny o określonej szerokości. Dzieli ona ciąg podany jako argument na linie o maksymalnej długości 75 znaków. Do rozdzielenia linii domyślnie jest używany znak \n. Oprócz argumentu określającego ciąg źródłowy funkcja posiada jeszcze trzy opcjonalne argumenty. Są to:

- liczba określająca maksymalną długość linii,
- ciąg znaków zastosowany do rozdzielenia linii
- argument podziału słów dłuższych niż zadeklarowana maksymalna długość linii.

```
1) $wordWrap=<<<abc
2) Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer finibus
   tincidunt metus, vitae ornare justo. Maecenas venenatis tortor ac tellus
   malesuada fringilla. Vestibulum quis nulla mi. Duis sit amet laoreet mi.
   Interdum et malesuada fames ac ante ipsum primis in faucibus. Proin
   mollis tincidunt facilisis. Fusce maximus tincidunt bibendum.
3) abc;
4) echo $wordWrap;
5) echo wordwrap($wordWrap, 40, "<br>\n", true);//bez dodatkowych dwóch
   argumentów nie działa, bo przegladarki nie interpretują \n ; aby długie
   słowo było podzielone musi być ostatni parametr true
```

Funkcje zmiany wielkości liter

Częstym działaniem na ciągach znakowych jest zamiana wszystkich liter na duże lub małe. Do zamiany wszystkich liter na duże służy funkcja: strtoupper(argument)

Ciąg podany w postaci argumentu zostanie zmodyfikowany w ten sposób, że wszystkie litery zostaną zamienione na duże.
Do zamiany wszystkich liter na małe służy funkcja: **strtolower(argument)**

Ciąg podany w postaci argumentu zostanie zmodyfikowany w ten sposób, że wszystkie litery zostaną zamienione na małe.

Jeżeli w ciągi znaków są znaki narodowe, to należy użyć funkcji
mb_ strtoupper(argument, [kodowanie]);

mb_ strtolower(argument, [kodowanie]);

```php
$str1="ala i as";
$strNaDuze=strtoupper($str1);
echo $strNaDuze;
$strNaMale=strtolower($strNaDuze);
echo "<br>";
echo $strNaMale;
```

ALA I AS
ala i as

```php
$str1="ala ąAĄEĘCććĆi as";
$strNaDuze=strtoupper($str1);
echo $strNaDuze;
$strNaMale=strtolower($strNaDuze);
echo "<br>";
echo $strNaMale;
```

//poniżej polskie litery nie zostały zmienione
ALA ąAĄEĘCććĆI AS
ala ąaĄeĘcććĆi as

```php
$str1="ala ąAĄEĘCććĆi as";
$strNaDuze=mb_strtoupper($str1);
echo $strNaDuze;
$strNaMale=mb_strtolower($strNaDuze);
echo "<br>";
echo $strNaMale;
```

//teraz „ogonki” są przekształcane

ALA ĄAĄEĘCĆĆĆI AS
ala ąaąeęcććći as

Kolejnymi funkcjami o podobnym działaniu są:

Funkcja ucfirst() zmodyfikuje podany argument w ten sposób, że pierwsza litera ciągu zostanie zamieniona na dużą literę.

Funkcja ucwords() zmodyfikuje ciąg podany jako argument w ten sposób, że wszystkie pierwsze litery wyrazów zostaną zamienione na duże litery.

Przykład:

```php
$str3="funkcja ucfirst() zmodyfikuje podany argument w ten sposób, że pierwsza litera ciągu zostanie zamieniona na dużą literę.";
echo ucfirst($str3);
echo "<br>";
$str3="funkcja ucwords() zmodyfikuje ciąg podany jako argument w ten sposób, że wszystkie pierwsze litery wyrazów zostaną zamienione na duże litery.";
echo ucwords($str3);
```

wynik działania:

Funkcja ucfirst() zmodyfikuje podany argument w ten sposób, że pierwsza litera ciągu zostanie zamieniona na dużą literę.
Funkcja Ucwords() Zmodyfikuje Ciąg Podany Jako Argument W Ten Sposób, że Wszystkie Pierwsze Litery Wyrazów Zostaną Zamienione Na Duże Litery.

Funkcje usuwania ciągu znaków

Do usunięcia z początku lub końca ciągu białych znaków można użyć jednej z trzech funkcji:

- **trim($str)** — usuwa podane znaki z początku i końca ciągu
- **ltrim($str)** — usuwa podane znaki z początku ciągu
- **rtrim($str)** — usuwa podane znaki z końca ciągu (tak samo działa funkcja **chop($str)**)

Wszystkie one mają taką samą konstrukcję:
nazwaFunkcji(„ciąg znakowy")

Usuwają one następujące znaki:

- **znak spacji** (kod 32),
- **znak tabulacji** (kod 9),
- **znak nowej linii** (kod 10),
- **znak powrotu karetki** (kod 13),
- **znak tabulacji pionowej** (kod 11),
- **znak o kodzie 0**.

Do usuwania innych znaków należy użyć podanych wyżej funkcji z dodatkowym parametrem w postaci:

```
Nazwa_funkcji("jakis tekst ze znakami do usuniecia: ", "znaki do usuniecia");
```

```
$wynik=trim("xxxxjakis tekst ze znakami do usuniecia: ", "x");
echo $wynik;
```

wynik: jakis tekst ze znakami do usuniecia:

funkcja strip_tags($tekst, [opcjonalnie-który znacznik nie powinien być usunięty]) – ze zmiennej $tekst zostaną usunięte znaczniki HTML, XHTML i PHP

```
echo strip_tags("Hello <b><i>world!</i></b>");
echo "<br>";
echo strip_tags("Hello <b><i>world!</i></b>","<b>");
echo "<br>";
echo strip_tags("Hello <b><i>world!</i></b>");
```

wynik działania:

Hello world!
Hello **world!**
Hello world!

Funkcje analizowania ciągów znaków

Do sprawdzenia długości ciągu znaków jest wykorzytywana funkcja **strlen($str).**

```
$napis="asdgfd";
$n=strlen($napis);
for($i=0; $i<$n; $i++){
    echo $napis[$i]."<br>";
}
```

Wynik działania:

a
s
d
g
f
d

Funkcje znajdowania podciągów w ciągu znaków

strstr(„przeszukiwany ciąg źródłowy", „szukany podciąg") – funkcja zwróci false, gdy podciąg nie zostanie znaleziony. Jeżeli szukany ciąg jest fragmentem ciągu źródowego, to funkcja zwróci fragment ciągu źródłowego od znalezionego podciągiu do jego końca. Funkcja strstr() rozróżnia wielkosc liter. Jeżeli wielkość liter nnie powinna mieć znaczenia, podczas przeszukiwania ciągu źródłowego należy użyć funkcji **stristr()**.

```
$dane="Jan Kowalski, ul. Polna 24, tel.  12345678";
$tel=strstr($dane, "tel");
echo $tel;
```

wynik działania: tel. 12345678

```
$dane="Jan Kowalski, ul. Polna 24, tel.  12345678";
$inneDane=strstr($dane, "PESEL");
echo $inneDane;
var_dump($inneDane);
```

wynik działania: bool(false)

```
$dane="Jan Kowalski, ul. Polna 24, tel.  12345678";
$ulica1=strstr($dane, "Ul.");
$ulica2=stristr($dane, "Ul.");
var_dump($ulica1);
echo "<br>";
echo $ulica2;
```

wynik działania:

bool(false)
ul. Polna 24, tel. 12345678

Podobne działania do funkcji strstr() ma funkcja strpos();
strpos(„przeszukiwany ciąg źródłowy", „szukany podciąg", [opcjonalnie-pozycja w ciągu źródłowym, od której rozpocznie się przeszukiwanie])
Jeżeli podciąg nie zostanie znaleziony-zwrócona zostanie wartośc false, w przeciwnym wypadku funkcja zwroci index okreslający miejsce znalezienia podciągu.
Przykład:

```php
$dane="asd php cds php sdfg";
$wynik1=strpos($dane,"php");
$wynik2=strpos($dane,"PHP");
$wynik3=strpos($dane,"abc");
var_dump($wynik1);
var_dump($wynik2);
var_dump($wynik3);
```

wynik:
int(4)
bool(false)
bool(false)

strrpos((„przeszukiwany ciąg źródłowy", „szukany podciąg", [opcjonalnie-pozycja w ciągu źródłowym, od której rozpocznie się przeszukiwanie]**)** funkcja znajdujaca ostatnie wystapienie stringa w innym stringu.

- strpos() - Finds the position of the first occurrence of a string inside another string (case-sensitive)
- strrpos() - Finds the position of the last occurrence of a string inside another string (case-sensitive)
- stripos() - Finds the position of the first occurrence of a string inside another string (case-insensitive)
- strripos() - Finds the position of the last occurrence of a string inside another string (case-insensitive)

funkcja substr() – zwraca część ciągu źródłowego
substr(„ciąg źródłowy", index wskazujący początek podciągu, [opcjonalnie- ilosc znaków]);

Jeżeli trzeci argument zostanie pominiety, zostanie zwrócony podciągod wskazanego indeksu do końca ciagu.
Jeżeli jako drugi argument zostanie podana wartość ujemna, pozycja indeksu będzie liczona od końca ciągu źródłowego

```
$pesel="55080319466";
$miesiac=substr($pesel,2,2);
echo $miesiac;
echo "<br>";
$plec=substr($pesel, -2, 1);
echo $plec%2==0?"k":"m";
```

wynik działania:

08
k

Funkcja strtok(„ciąg źródłowy”, „ciąg znaków rozdzielających”) pozwala na podzielenie ciągu znaków na podciągi. Funkcja ta przy pierwszym wywołaniu zwraca pierwszy wydzielony ciąg i zapamietuje wydzielony ciąg w pamieci podręcznej. Przy kolejnym wywołaniu funkcja zwraca kolejne ciagi. Jeżeli zostanie osiągniety konniec ciągu, funkcja zwraca wartość false.

Podziel ciąg jeden po drugim:

W poniższym przykładzie należy zauważyć, że tylko pierwsze wywołanie strtok() używa argumentu ciągu. Po pierwszym wywołaniu ta funkcja potrzebuje tylko argumentu separatora, ponieważ śledzi, gdzie się znajduje w bieżącym ciągu. Aby otrzymać nowy ciąg, ponownie należy wywołać strtok() z separatorem ciągu:

```
$dane="Jan Kowalski, ul. Polna 13, 44-113 Gliwice, tel. 12345432";
$sep=",";
$dziel=strtok($dane,$sep);
echo $dziel;
echo "<br>";
$dziel=strtok($sep);
echo $dziel;
echo "<br>";
$dziel=strtok($sep);
echo $dziel;
echo "<br>";
$dziel=strtok($sep);
echo $dziel;
var_dump( $dziel);
echo "<br>";
$dziel=strtok($sep);
echo $dziel;
var_dump( $dziel);
```

wynik działania:

Jan Kowalski
ul. Polna 13

44-113 Gliwice
tel. 12345432string(14) " tel. 12345432"
bool(false)

zdecydowanie lepiej w pętli:

```
$dane="Jan Kowalski, ul. Polna 13, 44-113 Gliwice, tel. 12345432";
$sep=",";
$dziel=strtok($dane,$sep);
while($dziel!==false){
    echo $dziel;
    echo "<br>";
    $dziel=strtok($sep);
}
```

Wynik działania:

Jan Kowalski
ul. Polna 13
44-113 Gliwice
tel. 12345432

Funkcje porównywania ciągów

Porównywanie ciągów znaków można wykonać za pomoca operatora porównania lub funkcji porównujących

strcmp(„ciąg 1”,”ciąg 2”)

zwraca wartość:

- mniejszą od 0, gdy ciąg 1< ciąg 2;
- równą 0, gdy ciagi są równe
- większą od 0, gdy ciąg 1> ciąg 2

funkcja rozróżnia wielkośćznaków

```
echo strcmp("Hello","Hello");
echo "<br>";
echo strcmp("A","H");
echo "<br>";
echo ord("H")," ", ord("A");
echo "<br>";
echo 72-65;
```

wynik dzialania:

0
-1
72 65
7

strcasecmp(„ciąg 1”,”ciąg 2”) – nie uwzglednia wielkości liter

funkcja

str_replace(*„co zamienić”,”na co zamienić” ,”string źródłowy,[opcjonalnie zmienna, która będzie pamietała ilosc zamian]*);

```
$str5="jakiś tekst php a tutaj inny  php i  tekst php";
$nowy=str_replace("php", "abcd", $str5, $n);
echo $nowy."  ".$n;
```

wynik:

jakiś tekst abcd a tutaj inny abcd i tekst abcd 3