CTIS 256 Web Technologies II

Note # 6

Serkan GENÇ

Regular Expression

- Regular expression (regex) is a powerful tool to define string patterns in a formal way.
- String patterns are phone number format, date, time, email addresses, urls, zip codes, and custom defined patterns such as the format of a flight ticket , serial no of an item.
- It is used in searching complex patterns and/or replacing a pattern with a new one.
- One line of “regex” is worth tens of lines of codes.
- It is supported by almost all languages (javascript, php, java, C, C++, etc.)
- There are different regex engines which are not fully compatible with each other. The engine used by PHP, Apache Web Server, Nginx is called PCRE. However, the differences are very slight.

```
// Regular expression Syntax:  DELIMETER pattern DELIMETER
// for example, /pattern/  or #pattern# or !pattern!
// DELIMITER is chosen by the programmer. Usually “/” is used as delimiter

// 1. Raw String Search :
//All strings that contain 'the'
preg_match( '/the/', 'The birds run away from them immediately') ;
preg_match( '/the/', 'them and therefore') ;

// 2. DOT operator
// dot is used as a placeholder for ANY ONE character.
preg_match( '/.the/', 'the man') ;
preg_match( '/.the/', 'It stops then') ;

preg_match( '/c.t/', 'there is a cat') ;
preg_match( '/c.t/', 'tshirt is made of cotton.') ;

//3. SQUARE BRACKET:  optional character selections.
// [0-9] represents one character that can be one of any digits
// [a-z], [A-Z], [a-zA-Z], [d-k,]  [a-zA-Z_]
// \d is a shorthand notation for [0-9]
// \w is for  [a-zA-Z0-9_]
// [^0-9] : a character that is NOT a digit. Caret shows negation.
// \b : word boundary, \s : whitespace
// \D: not digit, \W: not word letter, \S: not whitespace

preg_match( '/\d\dTR\d\d/', 'ticket id is 34TR45678' ) ;
preg_match( '/\d\dTR\d\d/', 'ticket id is 34TR4K5678' ) ;

preg_match( '/\bthe\b/' , 'and their solution') ;
preg_match( '/\bthe\b/i' , 'right.The important') ; // i modifier for case insensitive
preg_match( '#\b[tT]he#' , 'and the regular expr') ;
preg_match( '#\b[tT]he#' , 'Therefore, it is ') ;
preg_match( '#\b[tT]he#' , 'important.The next one ') ;
preg_match( '#\b[tT]he#' , 'mathematics') ;
```

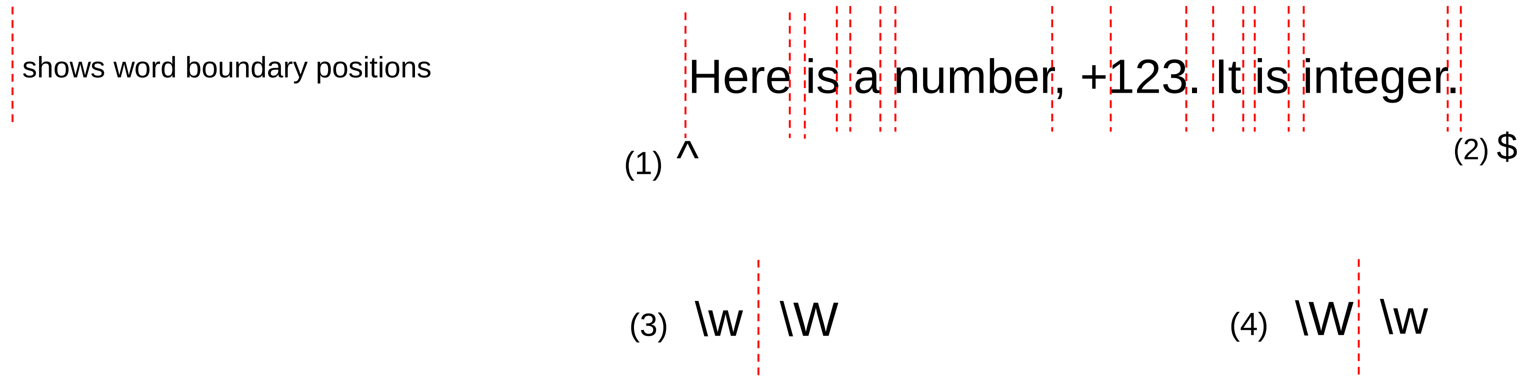
```
// 4. QUESTION MARK
// ? : optional
preg_match( '#\b\d\d?:\d\d?\b#' , 'time is 12:45') ;
preg_match( '#\b\d\d?:\d\d?\b#' , 'time is  2:13') ;
preg_match( '#\b\d\d?:\d\d?\b#' , 'time is  2:1') ;
preg_match( '#\b\d\d?:\d\d?\b#' , 'time is  122:13') ;

// 5. PLUS :  + influence the preceding charater, and it means one or more repetition
preg_match( '/VISA\d+END/' , 'Your code is VISA123432END') ;
preg_match( '/VISA\d+END/' , 'Your code is VISA12END123') ;
preg_match( '/VISA\d+END/' , 'Your code is VISAEND') ;

// 6. STAR : * means zero or more repetition
preg_match( '/VISA\d+END/' , 'Your code is VISA123432END') ;
preg_match( '/VISA\d*END/' , 'Your code is VISAEND') ;

// 7. CURLY BRACES : {n} means n repetition, {n1, n2} means at least n1, and at most n2 repetitions.
preg_match( '#\b\d{1,2}-\d{1,2}-\d{4}\b#' , 'the flight date is 12-1-2013 on Monday.') ;

// 8. VERICAL BAR : | shows 'or' operation
preg_match('/cat|dog/i', 'do you have a cat in your house') ;
preg_match('/cat|dog/i', 'do you want to eat a hotdog') ;

// 8. PARANTHESIS : () shows the grouping and capturing.
preg_match('/\b(\d\d)+\b/', 'The numbers are 2356') ; // even number of digits.
preg_match('/\b(\d\d)+\b/', 'The numbers are 23567') ;

// find references with square brackets or paranthesis in a sting.
preg_match('/\(\d{1,3}\)|\[\d{1,3}\]/', "You can see [13] for further") ;
preg_match('/\(\d{1,3}\)|\[\d{1,3}\]/', "You can see (13) for further") ;

preg_match('!\b\d{1,2}-\d{1,2}-(\d{2}|\d{4})\b!', 'date is 12-1-13' ) ;
preg_match('!\b\d{1,2}-\d{1,2}-(\d{2}|\d{4})\b!', 'date is 12-1-2013' ) ;

preg_match( '/\w+@(\w+\.){1,3}(com|tr)/i', ' my email is sgenc@bilkent.edu.tr') ;
preg_match( '/\w+@(\w+\.){1,3}(com|tr)/i', ' my email is sgenc@hotmail.de') ;
preg_match( '/\w+@(\w+\.){1,3}(com|tr)/i', ' my email is sgenc@hotmail.com') ;
```

```
//9. CARET : ^ means starting with.
preg_match('/^\d{2}\b/', '12 lemon is ...') ; //starting with two digits
preg_match('/^\d{2}\b/', '124 lemon is ...') ;

// 10. DOLLAR SIGN : $ means ending with.
preg_match('/tion$/i', 'caption') ;
preg_match('/tion$/i', 'caption.') ;

// EXACT MATCHING : Using ^ and $
preg_match('/^\d{3}$/', 'This is 123') ;
preg_match('/^\d{3}$/', '123') ;
preg_match('/^\d{3}$/', '1234') ;

preg_match( '#^\d{1,2}-\d{1,2}-\d{4}$#' , 'the flight date is 12-1-2013 on Monday.') ;
preg_match( '#^\d{1,2}-\d{1,2}-\d{4}$#' , '12-1-2013') ;

// BACKREFERENCE
preg_match( '#^\d{1,2}(-|\.)\d{1,2}(-|\.)\d{4}$#' , '12-1-2013') ;
preg_match( '#^\d{1,2}(-|\.)\d{1,2}(-|\.)\d{4}$#' , '12-1.2013') ; // Not working

// here \1 represents the content of the first group (hypen or dot).
preg_match( '#^\d{1,2}(-|\.)\d{1,2}\1\d{4}$#' , '12-1.2013') ;
preg_match( '#^\d{1,2}(-|\.)\d{1,2}\1\d{4}$#' , '12.1.2013') ;
preg_match( '#^\d{1,2}(-|\.)\d{1,2}\1\d{4}$#' , '12-1-2013') ;
```

Word Boundary

- **\b** shows word boundary positions.
- It is not a character.
- (1) Start of the line/string is a word boundary position, ^
- (2) End of the line/string is a word boundary position, $
- (3) \w to \W transition is a word boundary position.
- (4) \W to \w transition is a word boundary position.

\w = [a-zA-Z0-9_]   \W = [^a-zA-Z0-9_]


shows word boundary positions
Here is a number, +123. It is integer.
(1) ^
(2) $
(3) \w \W
(4) \W \w

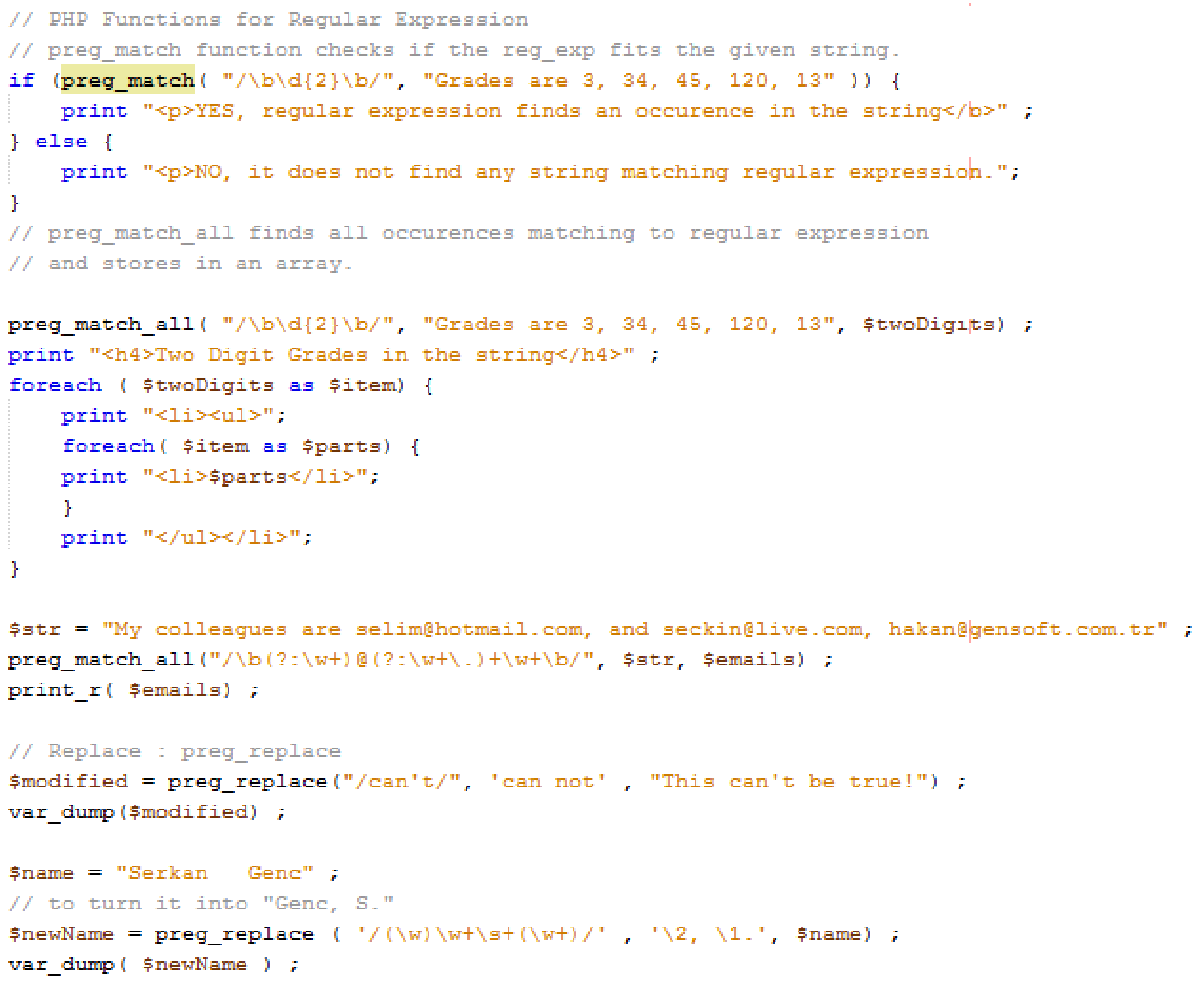
Regex functions in PHP

preg_match(string $pattern, string $subject) : int

returns 1 if the pattern matches the subject, 0 otherwise.

preg_match_all(string $pattern, string $subject, [array &$matches]) : int
matches all occurrences of the "*pattern*" within the "*subject*" and stores them in “matches” array. It returns the number of matches.

preg_replace(string $pattern, string $replacement, mixed $subject) : string
replaces matching strings to “pattern” with matching strings to “replacement” in string “subject” and it returns the modified one. Original string does not change.



```php
// PHP Functions for Regular Expression
// preg_match function checks if the reg_exp fits the given string.
if (preg_match( "/\b\d{2}\b/", "Grades are 3, 34, 45, 120, 13" )) {
    print "<p>YES, regular expression finds an occurence in the string</b>" ;
} else {
    print "<p>NO, it does not find any string matching regular expression.";
}
// preg_match_all finds all occurences matching to regular expression
// and stores in an array.

preg_match_all( "/\b\d{2}\b/", "Grades are 3, 34, 45, 120, 13", $twoDigits) ;
print "<h4>Two Digit Grades in the string</h4>" ;
foreach ( $twoDigits as $item) {
    print "<li><ul>";
    foreach( $item as $parts) {
    print "<li>$parts</li>";
    }
    print "</ul></li>";
}

$str = "My colleagues are selim@hotmail.com, and seckin@live.com, hakan@gensoft.com.tr" ;
preg_match_all("/\b(?:\w+)@(?:\w+\.)+\w+\b/", $str, $emails) ;
print_r( $emails) ;

// Replace : preg_replace
$modified = preg_replace("/can't/", 'can not' , "This can't be true!") ;
var_dump($modified) ;

$name = "Serkan   Genc" ;
// to turn it into "Genc, S."
$newName = preg_replace ( '/(\w)\w+\s+(\w+)/' , '\2, \1.', $name) ;
var_dump( $newName ) ;
```