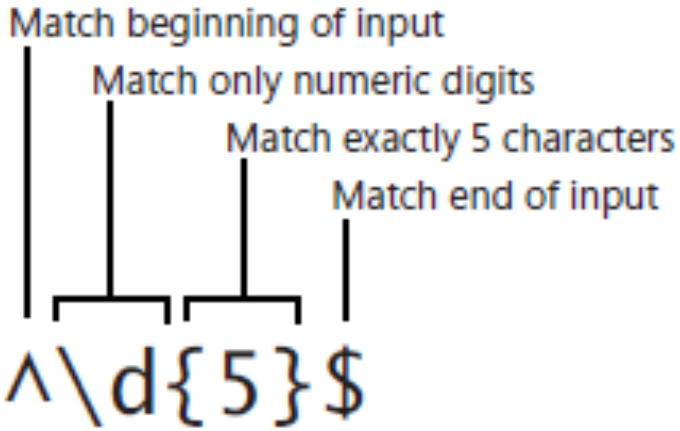
Regular expressions

Regular expressions

- A regular expression is a set of characters that can be compared to a string to determine whether the string meets specified format requirements.
- ***System.Text.RegularExpressions* namespace**
- ***System.Text.RegularExpressions.Regex.IsMatch*** static method

Example

```
 string pattern = "^\\d{5}$";
            string input = "12345";
            if (Regex.IsMatch(input, pattern))
                Console.WriteLine("Input matches regular
expression.");
            else
                Console.WriteLine("Input DOES NOT match
regular expression.");
```



If you remove the first character from the regular expression, you drastically change the meaning of the pattern. The regular expression "\d{5}$" will still match valid five-digit numbers, such as "12345". However, it will also match the input string "abcd12345"

Handicaps

Regular expressions are an extremely efficient way to check user input, **BUT:**

- Regular expressions are difficult to create unless you are extremely familiar with the format.
- Creating regular expressions **can be** confusing, but reading regular expressions **definitely is.**

Some issues

- When developing in C#, you should begin every regular expression with an **@** so that backslashes are treated
- Notice that regular expressions are case-sensitive
- Often, capitalized characters have the opposite meaning of lowercase characters
- Don’t even try to memorize every regular expression code

Examples – “^” , “$”

- The regular expression “abc” matches the strings “abc”, “abcde”, or “yzabc” because each of the strings contains the regular expression. No **“*”** is necessary.
- To match text beginning at the first character of a string, start the regular expression with a “^” symbol. So, the regular expression “^abc” matches the strings “abc” and “abcde”, but it does not match “yzabc”.
- To match text that ends at the last character of a string, place a “$” symbol at the end of the regular expression.
- For example, the regular expression “abc$” matches “abc” and “yzabc”, but it does not match “abcde”.
- To exactly match a string, include both “^” and “$”. For example, “^abc$” only matches “abc” and does not match “abcde” or “yzabc”.

Using wildcards – “*”, “+”

- The **“*”** symbol matches the preceding character *zero or more times*.
- For example, **“to*n”** matches **“ton”, “tooon”,** or **“tn”.**

- The **“+”** symbol works similarly, but it must match *one or more times*.
- For example, **“to+n”** matches **“ton”** or **“tooon”,** but not **“tn”.**

Using wildcards – “{}”

- To match a specific number of repeated characters, use **“{*n*}”** where *n* is a digit.
- For example, **“to{3}n”** matches **“tooon”** but not **“ton”** or **“tn”.**

- To match a range of repeated characters, use **“{*min,max*}”.** For example, **“to{1,3}n”** matches **“ton”** or **“tooon”** but not **“tn”** or **“toooon”.**

- To specify only a minimum, leave the second number blank.
- For example, **“to{3,}n”** requires three or more consecutive **“o”** characters.

Using wildcards – “?”, “.”

- To make a character optional, use the **“?”** symbol.
- For example, **“to?n”** matches **“ton”** or **“tn”,** but not **“tooon”.**

- To match any single character, use **“.”**.
- For example, **“to.n”** matches **“totn”** or **“tojn”** but not **“ton”** or **“tn”.**

Using wildcards – “[ ]”

- To match one of several characters, use brackets.
- For example, **“to[ro]n”** would match **“toon”** or **“torn”,** but not **“ton”** or **“toron”.**

- You can also match a range of characters.
- For example, **“to[o-r]n”** matches **“toon”, “topn”, “toqn”,** or **“torn”** but would not match **“toan”** or **“toyn”.**
- **[a-z] –** range of characters

x | y

- Matches either x or y.
- For example, **“z|food”** matches **“z”** or **“food”. “(z|f)ood”** matches **“zood”** or **“food”.**

Characters used in RE

Character	Description
\d	Matches a digit character. Equivalent to "[0-9]".
\D	Matches a nondigit character. Equivalent to "[^0-9]".
\s	Matches any white-space character, including Space, Tab, and form-feed. Equivalent to "[\f\n\r\t\v]".
\S	Matches any non-white-space character. Equivalent to "[^ \f\n\r\t\v]".
\w	Matches any word character, including underscore. Equivalent to "[A-Za-z0-9_]".
\W	Matches any nonword character. Equivalent to "[^A-Za-z0-9_]".

Matching a group of characters

- To match a group of characters, surround the characters with parentheses.
- For example, **"foo(loo){1,3}hoo"** would match **"fooloohoo"** and **"foolooloolooloohoo"** , but not **"foohoo"** or **"foolololohoo".**

- Similarly, **"foo(loo|roo)hoo"** would match either **"fooloohoo"** or **"fooroohoo".** You can apply any wildcard or other special character to a group of characters.

RE options

- You can modify a regular expression pattern with options that affect matching behavior.
- They can be specified in the options parameter in the Regex(pattern, options) constructor
- IgnoreCase, Multiline, Singleline…

Example

- abc
  def
  ghi
- If this text file is copied to the **s** String, the following method returns false because **"def"** is not at both the beginning and end of the string:

  Regex.IsMatch(s, "^def$")

- But the following method returns *true* because the ***RegexOptions.Multiline*** option enables the **"^"** symbol to match the beginning of a line, and it enables the **"$"** symbol to match the end of a line:

Regex.IsMatch(s, "^def$", *RegexOptions.Multiline*)

How to Extract Matched Data

1. Create a regular expression, and enclose in parentheses the pattern to be matched.
2. Create an instance of the **System.Text.RegularExpressions.Match** class using the static **Regex.Match** method.
3. Retrieve the matched data by accessing the elements of the **Match.Groups** array.

```
string input = "Company Name: Contoso, Inc.";
Match m = Regex.Match(input, @"Company Name: (.*$)");
Console.WriteLine(m.Groups[1]);
```

Example

- Suppose you want to distinguish between a phone number and invalid data.
- Phone can have one of the following formats:
  - (555)555–1212
  - (555) 555–1212
  - 555-555–1212
  - 5555551212

@"^\(?\d{3}\)?[\s\-]?\d{3}\-?\d{4}$"

Example (cont.)

(555)555-1212
(555) 555-1212
555-555-1212
5555551212

@"^\(?\d{3}\)?[\s\-]?\d{3}\-?\d{4}$"

- **^** Matches the beginning of the string.
- **\(?** Optionally matches an opening parenthesis. The parenthesis is preceded with a backslash, because the parenthesis is a special character in regular expressions. The following question mark makes the parenthesis optional.
- **\d{3}** Matches exactly three numeric digits.
- **\)?** Optionally matches a closing parenthesis. The parenthesis is preceded with a backslash because the parenthesis is a special character in regular expressions. The following question mark makes the parenthesis optional.
- **[\s\-]?** Matches either a space ("\s") or a hyphen ("\-") separating the area code from the rest of the phone number. The following question mark makes the space or hyphen optional.
- **\d{3}** Matches exactly three numeric digits.
- **\-?** Optionally matches a hyphen.
- **\d{4}$** Requires that the string end with four numeric digits.

Summary

- Regular expressions enable you to determine whether text matches almost any type of format. Regular expressions support dozens of special characters and operators. The most commonly used are **"^"** to match the beginning of a string, **"$"** to match the end of a string, **"?"** to make a character optional, **"."** to match any character, and **"*"** to match a repeated character.

- To match data using a regular expression, create a pattern using groups to specify the data you need to extract, call ***Regex.Match*** to create a ***Match*** object, and then examine each of the items in the ***Match.Groups*** array.

an's Tools
xpal.com

Blocking site with

Related: Original Regexpal

Regular Expression

```
/unblocked-*games/ig
```

Test String

```
https://sites.google.com/site/unblockedgames66/
https://sites.google.com/site/unblockedgames77/
https://sites.google.com/site/unblockedgames4me/
https://sites.google.com/site/unblockedgamesvevo/
https://online-unblocked-games.weebly.com/
http://unblockedgames76.weebly.com/
```

Substitution