Page1. **대소문자를 구별함** : Regular expressions are case sensitive. Therefore Case 1 will find the specified text, but Case 2 will not.

Source

Hello, world!

Case 1

Regular Expression: Hello
First match: Hello, world!
All matches: Hello, world!

Case 2

Regular Expression: hello
First match: Hello, world!
All matches: Hello, world!

Page2. **공백 문자를 포함** : Each character inside the search pattern is significant including whitespace characters (space, tab, new line).

Source

Hello, world!

Case 1

Regular Expression: Hello, world
First match: Hello, world!
All matches: Hello, world!

Case 2

Regular Expression: Hello, world
First match: Hello, world!
All matches: Hello, world!

Page3. **^는 줄의 시작, $는 줄의 마지막을 의미** : Some characters have special meanings. Character ^ matches the beginning of the line while dollar sign $ the end of the line

Source

who is who

Case 1

Regular Expression: ^who
First match: who is who
All matches: who is who

Case 2

Regular Expression: who$
First match: who is who
All matches: who is who

Page4. **리터럴** : If literal value of a special character is required, it must be escaped with a backslash ₩. Case 1 does not match anything as both characters

a special, Case 2 matches all $, Case 3 matches $ only if it is the first and Case 4 the last character. Backslash has special meaning and must be also escaped for literal use.

Source

12 \-\ 25

Case 1

Regular Expression: ^$
First match: 12 \-\ 25
All matches: 12 \-\ 25

Case 2

Regular Expression: \$
First match: 12 \-\ 25
All matches: 12 \-\ 25

Case 3

Regular Expression: ^\$
First match: 12 \-\ 25
All matches: 12 \-\ 25

Case 4

Regular Expression: \$$
First match: 12 \-\ 25
All matches: 12 \-\ 25

Case 5

Regular Expression: \\
First match: 12 \-\ 25
All matches: 12 \-\ 25

Page5. **.(point)는 모든 글자를 의미** : Point . matches any character.

Source

Regular expressions are powerful!!!

Case 1

Regular Expression: .
First match: Regular expressions are powerful!!!
All matches: **Regular expressions are powerful!!!**

Case 2

Regular Expression: .....
First match: **Regular** expressions are powerful!!!
All matches: **Regular expressions are powerf**ul!!!

Page6. .(point)의 리터럴 : The point must be escaped if literal meaning is required.

Source

O.K.

Case 1

Regular Expression: .
First match: O.K.
All matches: O.K.

Case 2

Regular Expression: \.
First match: O.K.
All matches: O.K.

Case 3

Regular Expression: \..\.
First match: O.K.
All matches: O.K.

Page7. **대괄호 안에 있는 글자 중 하나만 있어도 매치(글자의 순서는 무관)** : Inside square brackets "[]" a list of characters can be provided. The expression matches if any of these characters is found. The order of characters is insignificant.

Source

How do you do?

Case 1

Regular Expression: [oyu]
First match: How do you do?
All matches: How do you do?

Case 2

Regular Expression: [dH].
First match: How do you do?
All matches: How do you do?

Case 3

Regular Expression: [owy][yow]
First match: How do you do?
All matches: How do you do?

Page8. **- 는 범위를 의미** : **A range of characters can be specified with [-] syntax**. Case 1 and Case 2 are equivalent. Several ranges can be given in one expression.

Source

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz 0123456789

Case 1

Regular Expression: **[C-K]**
First match: ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz 0123456789
All matches: ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz 0123456789

Case 2

Regular Expression: **[CDEFGHIJK]**
First match: ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz 0123456789
All matches: ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz 0123456789

Case 3

Regular Expression: **[a-d]**
First match: ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz 0123456789
All matches: ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz 0123456789

Case 4

Regular Expression: **[2-6]**
First match: ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz 0123456789
All matches: ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz 0123456789

Case 5

Regular Expression: **[C-Ka-d2-6]**
First match: ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz 0123456789
All matches: ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz 0123456789

Page9. **^는 not을 의미** : If a character class starts with ^, then specified characters will not be selected.

Source

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz 0123456789

Case 1

Regular Expression: [^CDghi45]

First match: ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz 0123456789

All matches: ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz 0123456789

Case 2

Regular Expression: [^W-Z]

First match: ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz 0123456789

All matches: ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz 0123456789

Page10. **|로 둘러쌓인 문자 중 아무거나 매칭됨** : Alternating text can be enclosed in parentheses and alternatives separated with |.

Source

Monday Tuesday Friday

Case 2

Regular Expression: (on|ues|rida)
First match: Monday Tuesday Friday
All matches: Monday Tuesday Friday

Case 2

Regular Expression: (Mon|Tues|Fri)day
First match: Monday Tuesday Friday
All matches: Monday Tuesday Friday

Case 3

Regular Expression: ..(id|esd|nd)ay
First match: Monday Tuesday Friday
All matches: Monday Tuesday Friday

Page11. ***은 0번 이상, +는 1번 이상, ?는 0번 또는 1번** : Quantifiers specify how many times a character can occur. Star * matches zero or more times, plus + once or more times and question mark ? zero or once.

Source

aabc abc bc

Case 1

Regular Expression: a*b
First match: aabc abc bc
All matches: aabc abc bc

Case 2

Regular Expression: a+b
First match: aabc abc bc
All matches: aabc abc bc

Case 3

Regular Expression: a?b
First match: aabc abc bc
All matches: aabc abc bc

Page12. ***의 다양한 사용예시** : Several examples of "*" quantifier

Source

-@- *** -- "*" -- *** -@-

Case 1

Regular Expression: .*
First match: -@- *** -- "*" -- *** -@-
All matches: -@- *** -- "*" -- *** -@-

Case 2

Regular Expression: -A*-
First match: -@- *** -- "*" -- *** -@-
All matches: -@- *** -- "*" -- *** -@-

Case 3

Regular Expression: [-@]*
First match: -@- *** -- "*" -- *** -@-
All matches: -@- *** -- "*" -- *** -@-

Page13. **+의 다양한 사용예시** : Several examples of "+" quantifier

Source

-@@@- * ** - - "*" -- * ** -@@@-

Case 1

Regular Expression: *+

First match: -@@@-***** ** - - "*" -- * ** -@@@-

All matches: -@@@-****** - - "***** -- ****** -@@@-

Case 2

Regular Expression: -@+-

First match: **-@@@-** * ** - - "*" -- * ** -@@@-

All matches: **-@@@-** * ** - - "*" -- * ** **-@@@-**

Case 3

Regular Expression: [^]+

First match: **-@@@-** * ** - - "*" -- * ** -@@@-

All matches: **-@@@-** ****** **- -** **"*"** **--** ****** **-@@@-**

Page14. **?의 다양한 사용예시** : Several examples of "?" quantifier

Source

--XX-@-XX-@@-XX-@@@-XX-@@@@-XX-@@-@@-

Case 1

Regular Expression: -X?XX?X

First match: -**-XX**-@-XX-@@-XX-@@@-XX-@@@@-XX-@@-@@-

All matches: -**-XX**-@**-XX**-@@**-XX**-@@@**-XX**-@@@@**-XX**-@@-@@-

Case 2

Regular Expression: -@?@?@?-

First match: **--**XX-@-XX-@@-XX-@@@-XX-@@@@-XX-@@-@@-

All matches: **--**XX**-@-**XX**-@@-**XX**-@@@-**XX-@@@@-XX**-@@-**@@-

Case 3

Regular Expression: [^@]@?@

First match: --XX**-@**-XX-@@-XX-@@@-XX-@@@@-XX-@@-@@-

All matches: --XX**-@**-XX**-@@**-XX**-@@**@-XX**-@@**@@-XX**-@@-@@-**

Page15. {}안의 숫자만큼 반복, {m,n}일 경우 최소 m번 최대n번 {m,} 최소 m번 : Curly brackets enable precise specification of character repetitions. {m} matches

precisely m times, {m,n} matches minimaly m times and maximaly n times and {m,}matches minimaly m times.

Source

One ring to bring them all and in the darkness bind them

Case 1

Regular Expression:	**.{5}**
First match:	**One r**ing to bring them all and in the darkness bind them
All matches:	**One ring to bring them all and in the darkness bind the**m

Case 2

Regular Expression:	**[els]{1,3}**
First match:	On**e** ring to bring them all and in the darkness bind them
All matches:	On**e** ring to bring th**em** a**ll** and in th**e** darkn**ess** bind th**em**

Case 3

Regular Expression:	**[a-z]{3,}**
First match:	One **ring** to bring them all and in the darkness bind them
All matches:	One **ring** to **bring** **them** **all** **and** in **the** **darkness** **bind** **them**

Page16. ***,+,?를 중괄호로 표현하기** : Quantifiers "*", "+", and "?" are special cases of the bracket notation. "*" is equivalent to {0,} , "+" to {1,} and "?" to {0,1}

Source

AA ABA ABBA ABBBA

Case 1

Regular Expression:	**AB*A**
First match:	**AA** ABA ABBA ABBBA
All matches:	**AA** **ABA** **ABBA** **ABBBA**

Case 2

Regular Expression:	**AB{0,}A**
First match:	**AA** ABA ABBA ABBBA
All matches:	**AA** **ABA** **ABBA** **ABBBA**

Case 3

Regular Expression:	**AB+A**
First match:	AA **ABA** ABBA ABBBA
All matches:	AA **ABA** **ABBA** **ABBBA**

Case 4

Regular Expression: AB{1,}A
First match: AA ABA ABBA ABBBA
All matches: AA ABA ABBA ABBBA

Case 5

Regular Expression: AB?A
First match: AA ABA ABBA ABBBA
All matches: AA ABA ABBA ABBBA

Case 6

Regular Expression: AB{0,1}A
First match: AA ABA ABBA ABBBA
All matches: AA ABA ABBA ABBBA

Page17. **?가 붙으면 가장 최소의 값으로 바뀜** : By default any subpattern matches as many times as possible. This behaviour is changed to matching the minimum number if quantifier is followed with the question mark. Compare "*" with "*?", "+" with "+?", and "?" with "??"

Source

One ring to bring them all and in the darkness bind them

Case 1

Regular Expression: r.*
First match: One ring to bring them all and in the darkness bind them
All matches: One ring to bring them all and in the darkness bind them

Case 2

Regular Expression: r.*?
First match: One ring to bring them all and in the darkness bind them
All matches: One ring to bring them all and in the darkness bind them

Case 3

Regular Expression: r.+
First match: One ring to bring them all and in the darkness bind them
All matches: One ring to bring them all and in the darkness bind them

Case 4

Regular Expression:	**r.+?**
First match:	One ring to bring them all and in the darkness bind them
All matches:	One ring to bring them all and in the darkness bind them

Case 5

Regular Expression:	**r.?**
First match:	One ring to bring them all and in the darkness bind them
All matches:	One ring to bring them all and in the darkness bind them

Case 6

Regular Expression:	**r.??**
First match:	One ring to bring them all and in the darkness bind them
All matches:	One ring to bring them all and in the darkness bind them

Page18. **~~₩~~w는 [A-z0-9_] 대신에 사용가능** : ~~₩~~w matches any word character (alphanumeric plus "_"). In some languages these letter abbreviations are not recognized. Use character classes ("[A-z0-9_]") instead.

Source

A1 B2 c3 d_4 e:5 ffGG77--__--

Case 1

Regular Expression:	**\w**
First match:	A1 B2 c3 d_4 e:5 ffGG77--__--
All matches:	A1 B2 c3 d_4 e:5 ffGG77--__--

Case 2

Regular Expression:	**\w***
First match:	A1 B2 c3 d_4 e:5 ffGG77--__--
All matches:	A1 B2 c3 d_4 e:5 ffGG77--__--

Case 3

Regular Expression:	**[a-z]\w***
First match:	A1 B2 c3 d_4 e:5 ffGG77--__--
All matches:	A1 B2 c3 d_4 e:5 ffGG77--__--

Case 4

Regular Expression: \w{5}
First match: A1 B2 c3 d_4 e:5 ffGG77--__--
All matches: A1 B2 c3 d_4 e:5 ffGG77--__--

Case 5

Regular Expression: [A-z0-9_]
First match: A1 B2 c3 d_4 e:5 ffGG77--__--
All matches: A1 B2 c3 d_4 e:5 ffGG77--__--

Page19. ₩W는 [^A-z0-9_] 대신 사용가능 : ₩W matches any non-word character (everything but alphanumeric plus "_"). Compare Case 1 and Case 2. It is equivalent to "[^A-z0-9_]".

Source

AS _34:AS11.23 @#$ %12^*

Case 1

Regular Expression: \W
First match: AS_34:AS11.23 @#$ %12^*
All matches: AS_34:AS11.23 @#$ %12^*

Case 2

Regular Expression: \w
First match: AS _34:AS11.23 @#$ %12^*
All matches: AS _34:AS11.23 @#$ %12^*

Case 3

Regular Expression: [^A-z0-9_]
First match: AS_34:AS11.23 @#$ %12^*
All matches: AS_34:AS11.23 @#$ %12^*

Page20. **₩s는 공백과 매치, ₩S는 공백이 아닌것과 매치** : ₩s matches white space characters: space, new line and tab. ₩S matches any non-whitespace character.

Source

Ere iron was found or tree was hewn,
When young was mountain under moon;
Ere ring was made, or wrought was woe,
It walked the forests long ago.

Case 1

Regular Expression:	**\s**
First match:	Ere iron was found or tree was hewn, When young was mountain under moon; Ere ring was made, or wrought was woe, It walked the forests long ago.
All matches:	Ere iron was found or tree was hewn, When young was mountain under moon; Ere ring was made, or wrought was woe, It walked the forests long ago.

Case 2

Regular Expression:	**\S**
First match:	Ere iron was found or tree was hewn, When young was mountain under moon; Ere ring was made, or wrought was woe, It walked the forests long ago.
All matches:	**Ere iron was found or tree was hewn, When young was mountain under moon; Ere ring was made, or wrought was woe, It walked the forests long ago.**

Page21. **₩d는 숫자를, ₩D는 숫자가 아닌 것을 의미** : ₩d matches any digit and ₩D anything else. Compare Case 1 and Case 2. Use "[0-9]"

Source

Ere iron was found or tree was hewn,
When young was mountain under moon;
Ere ring was made, or wrought was woe,
It walked the forests long ago.

Case 1

Regular Expression:	**\s**
First match:	Ere iron was found or tree was hewn, When young was mountain under moon; Ere ring was made, or wrought was woe, It walked the forests long ago.
All matches:	Ere iron was found or tree was hewn, When young was mountain under moon; Ere ring was made, or wrought was woe, It walked the forests long ago.

Case 2

Regular Expression: \S

First match: Ere iron was found or tree was hewn, When young was mountain under moon; Ere ring was made, or wrought was woe, It walked the forests long ago.

All matches: Ere iron was found or tree was hewn, When young was mountain under moon; Ere ring was made, or wrought was woe, It walked the forests long ago.

Page22. **₩b는 단어의 첫 글자를 의미** : ₩b matches a word boundary. A word boundary (₩b) is defined as a spot between two characters that has a ₩w on one side of it and a ₩W on the other side of it (in either order).

Source

Ere iron was found or tree was hewn,
When young was mountain under moon;
Ere ring was made, or wrought was woe,
It walked the forests long ago.

Case 1

Regular Expression: \b.

First match: Ere iron was found or tree was hewn, When young was mountain under moon; Ere ring was made, or wrought was woe, It walked the forests long ago.

All matches: Ere iron was found or tree was hewn, When young was mountain under moon; Ere ring was made, or wrought was woe, It walked the forests long ago.

Case 2

Regular Expression: .\b

First match: Ere iron was found or tree was hewn, When young was mountain under moon; Ere ring was made, or wrought was woe, It walked the forests long ago.

All matches: Ere iron was found or tree was hewn, When young was mountain under moon; Ere ring was made, or wrought was woe, It walked the forests long ago.

Page22. **₩B는 단어의 첫 글자를 제외한 글자를 의미** : ₩B matches a non (word boundary). A word boundary (₩b) is defined as a spot between two characters that has a ₩w on one side of it and a ₩W on the other side of it (in either order).

Source

Ere iron was found or tree was hewn,
When young was mountain under moon;
Ere ring was made, or wrought was woe,
It walked the forests long ago.

Case 1

Regular Expression: \B.

First match: Ere iron was found or tree was hewn, When young was mountain under moon; Ere ring was made, or wrought was woe, It walked the forests long ago.

All matches: Ere iron was found or tree was hewn, When young was mountain under moon; Ere ring was made, or wrought was woe, It walked the forests long ago.

Case 2

Regular Expression: .\B

First match: Ere iron was found or tree was hewn, When young was mountain under moon; Ere ring was made, or wrought was woe, It walked the forests long ago.

All matches: Ere iron was found or tree was hewn, When young was mountain under moon; Ere ring was made, or wrought was woe, It walked the forests long ago.

Page23. **₩A는 문장의 시작 단어만 매치(^는 모든 줄마다, ₩A는 문장에 한 개), ₩Z는 문장의 마지막 단어** : ₩A matches the beginning of string. It is similar to ^, but ^ will match after each newline, if multiline strings are considered. Similarly, ₩Z matches only at the end of the string or before newline at the end of it. It is similar to $, but $ will match before each newline.

Case 1

Regular Expression: \A...

First match: Ere iron was found or tree was hewn, When young was mountain under moon; Ere ring was made, or wrought was woe, It walked the forests long ago.

All matches: Ere iron was found or tree was hewn, When young was mountain under moon; Ere ring was made, or wrought was woe, It walked the forests long ago.

Case 2

Regular Expression: ...\Z

First match: Ere iron was found or tree was hewn, When young was mountain under moon; Ere ring was made, or wrought was woe, It walked the forests long ago.

All matches: Ere iron was found or tree was hewn, When young was mountain under moon; Ere ring was made, or wrought was woe, It walked the forests long ago.

Page25. (?=<pattern>) will look ahead if the pattern exists, but will not include it in the hit.

Source

AAAX---aaax---111

Case 1

Regular Expression: \w+(?=X)

First match: AAAX---aaax---111

All matches: AAAX---aaax---111

Case 2

Regular Expression: \w+

First match: AAAX---aaax---111

All matches: AAAX---aaax---111

Case 3

Regular Expression: \w+(?=\w)

First match: AAAX---aaax---111

All matches: AAAX---aaax---111

Page26. (?!<pattern>) will look ahead if the pattern exists. If it does there will be no hit.

Source

AAAX---AAA

Case 1

Regular Expression: AAA(?!X)

First match: AAAX---AAA

All matches: AAAX---AAA

Case 2

Regular Expression: AAA

First match: AAAX---AAA

All matches: AAAX---AAA