

# Contents

basic setup . . . . .	1
GREEDY and BACKTRACK . . . . .	2
AAA (fix) . . . . .	2
judy followed by digit (Fails) . . . . .	2
trim extra space [ nothing works here] . . . . .	3
LOOKAROUNDS . . . . .	4
FIND a st b immediately preceeds . . . . .	4

## LOOK AROUNDS : do not capture 4

```
## NOT A SCRIPT FILE

# =====
# PURPOSE: COLLECT EXAMPLES of GREP and regex (-P = Perl)
# USAGE:
#           - NOT A SCRIPT
#           - USE      neovim terminal
#           - USE ,tl to run each line in terminal
./001.... >> file.txt
#
# file <- "/home/jim/code/zsh_project/ZSH_SH_FILES/001_grep_regex_P_examples.md"
# :vsplit term://zsh
#
# =====
#### REF

- !so 3512471 - non-capture (?:)
```

## basic setup

```
str="hello"
line_break="-----"
regex="h"
echo $str | grep -E "$regex" -
regex='\d(?:=x) '

## GREP --color=always is now in the alias for grep
## SEE $ZSH alias
## hmmm, seems need to put it in here ... ???

alias grep='grep --color=always'

echo ${line_break}
echo "so 3512471"

str='animal=cat,dog,cat,'
echo $str
regex='(?:animal)(?:=)(\w+)(,)(?:.*)\1\2'
echo ${regex}
echo $str | grep -P $regex
echo 'animal=dog,cat,dog,deer,dog,' | grep -P $regex
```

## GREEDY and BACKTRACK

Consider the String <A tasty apple> (excluding the angle brackets)  
And the pattern <.\*apple> (again exclude the angle brackets)

```
initially, .* portion makes the entire string, but then can't make apple
str="A tasty apple"
regex='.*apple'
echo $str | grep -P $regex
echo $str | grep -P ".*"
```

### AAA (fix)

```
str="AAA"
echo $str | grep -P "A+"
echo $str | grep -P "(A+)."
```

  

```
echo $str | grep -E "(A+)."
```

  

```
# greedy
str="AAA"
regex='A*'
echo $str | grep -P $regex
```

  

```
regex='A*?'
echo $str | grep -P $regex
```

  

```
regex='A+?'
echo $str | grep -P $regex
```

  

```
str='-- comment'
regex='--(?:.*\S)?'
echo $str | grep -P $regex
```

### judy followed by digit (Fails)

```
### This ENDS dquote>
echo "
```

  

```
# Works
echo "judy is 3" | grep -P '\d'
```

  

```
# new topic
echo "Find Judith followed somewhere by digit"
```

  

```
## ALL FAIL
echo "Judith is 10 on scale of 10!" | grep -P "Judith(=.*?[0-9]\S)"
echo "Judith is 10 on scale of 10!" | grep -P 'Judith(=.*?[0-9])'
echo ""
```

  

```
echo "\"'"
echo "\""
```

```

regex='Judith(?.*[0-9])'
echo ${regex}

## FAIL
echo "Judith is 10 on scale of 10!" | grep -P ${regex}

echo ${line_break}
echo "Find digits"
echo "Judith is 10 on scale of 10!" | grep -P '\d'
echo "Judith is 10 on scale of 10!" | grep -P \d
echo "Judith is 10 on scale of 10!" | grep -P "\d"

echo "

## Works
echo "digit IF followed by somewhere by x"
echo "3 ....a ...a" | grep -P '\d(?.*x)'
echo "3 ....x ...x" | grep -P '\d(?.*x)'

## Works
echo ${line_break}
echo "digit IF followed immediately by x"

## compare:
echo "A: 3 ....x ...x" | grep -P '\d(?.*x)'
echo "B: ....3x ...x" | grep -P '\d(?.*x)'

## WORKS
echo ${line_break}
echo "Match x and digits in either order; C does not match"
echo "B works because followed by is zero width; sees x, so it meets criteria"
echo "C does not"

echo "A: ...x3" | grep -P '^(?.*x).*\d'
echo "B: ...3x" | grep -P '^(?.*x).*\d'
echo "C: ...3" | grep -P '^(?.*x).*\d'

echo ${line_break}
echo "to be or not to be" | grep -P '(to)(be) or not \1 \2'
echo "to be or not to be" | grep -P '(to)(be)(?:or not) \1 \2'

echo "2021-12-13" | grep -P '(\d{4})-(\d{2})'
echo "2021-12-13" | grep -P '(\d{4})-\d{2}\1'

echo "2021-12-13" | grep -P '(\d{4})-(\d{2})-(\d{2})\3' #MONTH\1'

```

### **trim extra space [ nothing works here]**

```

regex='^[ \t]+|[ \t]+$'
regex='^[ \s]+|[ \s]+$'
echo $regex

echo " abcde" | grep -P $regex
echo "abcde " | grep -P $regex

# Fails

```

```
echo " abcde " | grep -P $regex
```

```
# X(?=Y) means X followed by Y
regex='--.*?(?=[^\r\n\S]*$)'
echo $regex
echo '-- this is a comment.' | grep -P $regex
```

## LOOKAROUNDS

```
echo ${line_break}
echo "a followed by c"
echo 'a(?=c)'
str=bacad
echo $str | grep -P 'a(?=c)'
```

```
echo "bacadc"
echo ${line_break}
echo "a NOT followed by c"
echo 'a(?!c)'
str=bacad
str=bacadc
echo $str | grep -P 'a(?!c)'
```

## FIND a st b immediately preceeds

```
echo ${line_break}
echo "a preceeded by by b"
reg='(?<=b)a'
echo $reg
str=bacad
echo $str | grep -P '(?<=b)a'
```

## LOOK AROUND : do not capture

```
echo " "
reg='(?<=costs)\s\d\.\d{2}'
echo "Penguin costs 2.99, and whale costs 5.99 but I only have 2.01 left"
echo "Penguin costs 2.99, and whale costs 5.99 but I only have 2.01 left" | grep -P $reg
```

```
echo " "
reg='\w+\s(?:=costs) '
echo "Penguin costs 2.99, and whale costs 5.99 but I only have 2.01 left"
echo "Penguin costs 2.99, and whale costs 5.99 but I only have 2.01 left" | grep -P $reg
```

```
echo ${line_break}
echo "a NOT preceeded by by b"
reg='(?<!b)a'
```

```
echo $reg
str=bacad
echo $str | grep -P '(?<!b)a'
```

```

echo ${line_break}
echo "Precendent"
reg='((ab|d)e)'
echo $reg
str=bacade
echo $str | grep -P '((ab|d)e)'

```

```

echo ${line_break}
echo "Alternative"
reg='(ab|d)e'
echo $reg
str=bacade
echo $str | grep -P '(ab|d)'

```

```

regex="(\w*)(.*)\\1"

```

```

regex='(\w*)(.*)\1'
echo "wordXABC" | grep -P ${regex}
echo "word;word" | grep -P ${regex}
echo "word;word1" | grep -P ${regex}

```

```

# matches "word" only
echo "word[]" | grep -P '(\w*)'

```

```

## In R, does what?
p.5 <- problems(pattern="(\\w*)(.*)\\1")

```