



BridgeLabz

Employability Delivered

Programming
Constructs -
Patterns

7. Patterns

Patterns or Regular expressions are special characters which help search data, matching complex patterns. Regular expressions are shortened as 'regexp' or 'regex'.

Basic Regex

Symbol	Descriptions
.	replaces any character
^	matches start of string
\$	matches end of string
*	matches up zero or more times the preceding character
\	Represent special characters
()	Groups regular expressions
?	Matches up exactly one character

Basic Regex Samples

mkdir pattern
Create a sample file

apple
bat
ball
ant
eat
pant
people
taste
sample (END)

ex1: Match pp

```
pattern $ cat sample | grep pp  
apple
```

ex2: Starts with a

```
pattern $ cat sample | grep ^a  
apple  
ant
```

ex3: Ends with ll

```
pattern $ cat sample | grep ll$  
ball
```

Basic Regex

Pattern	Description
*	Match zero or more characters
?	Match any single character
[...]	Match any of the characters in a set
?(patterns)	Match zero or one occurrences of the patterns (extglob)
*(patterns)	Match zero or more occurrences of the patterns (extglob)
+(patterns)	Match one or more occurrences of the patterns (extglob)
@(patterns)	Match one occurrence of the patterns (extglob)
!(patterns)	Match anything that doesn't match one of the patterns (extglob)

Basic Regex Samples

```
$ touch a.jpg b.gif c.png d.pdf e.pdf ee.pdf
```

```
$ ls  
a.jpg b.gif c.png d.pdf ee.pdf
```

```
$ ls *.jpg  
a.jpg
```

```
$ ls ?.pdf  
d.pdf
```

```
$ ls [ab]*  
a.jpg b.gif
```

```
$ shopt -s extglob # turn on extended globbing
```

```
$ ls ?(*.jpg|*.gif)  
a.jpg b.gif
```

```
$ ls !(*.jpg|*.gif) # not a jpg or a gif  
c.png d.pdf ee.pdf
```

```
$ ls *.pdf  
ee.pdf e.pdf .pdf
```

```
$ ls ?(e).pdf # zero or one "e" allowed  
e.pdf .pdf
```

```
$ ls *(e).pdf # zero or more "e"s allowed  
ee.pdf e.pdf .pdf
```

```
$ ls +(e).pdf # one or more "e"s allowed  
ee.pdf e.pdf
```

```
$ ls @(e).pdf # only one e allowed  
e.pdf
```

Basic Regex Samples

```
#!/usr/local/bin/bash -x
shopt -s extglob # turn on extended globbing
echo "Enter word ending with thing"
read word
if [[ $word == +(some|any)thing ]];
then
    echo yes;
else
    echo no;
fi
patternTestV1.sh (END)
```

```
pattern $ ./patternTestV1.sh
+ shopt -s extglob
+ echo 'Enter word ending with thing'
Enter word ending with thing
+ read word
something
+ [[ something == +(some|any)thing ]]
+ echo yes
yes
```

Regex Explained

Valid & Invalid Ones

111 \Rightarrow Invalid

1aqa \Rightarrow Valid

aa11 \Rightarrow Invalid

bcc \Rightarrow Valid

1a1a1b \Rightarrow Invalid

(a b b 23a \Rightarrow Valid

Regex Explained – Derive the Rules

- 1 → Valid ones have ^{min} 8 consecutive characters
- 2 → Preceded by zero or more numbers & consecutive characters till the
- 3 → Succeeded by zero or more numbers & characters

Regex Explained – Patterns

Patterns

1:) `[a-zA-Z]{3,}`

2:) `^[0-9]*`

3:) `[0-9 a-zA-Z]*$`

So the combine pattern is

"`^([0-9]*[a-zA-Z]){3,}`
`[0-9 a-zA-Z]*$`"

Regex Explained – Execution

```
#1: String of 3 consecutive characters with no special characters
pat="^([0-9]*[a-zA-Z]){3,}[0-9]*$"
any="aaa1"
if [[ $any =~ $pat ]]; then echo yes; else echo no; fi
```

```
+ pat='^([0-9]*[a-zA-Z]){3,}[0-9]*$'
+ any=aaa1
+ [[ aaa1 =~ ^([0-9]*[a-zA-Z]){3,}[0-9]*$ ]]
+ echo yes
```



PIN Code UC 1

The Postal Index Number (PIN) or PIN Code is a 6 digit code of Post Office numbering used by India Post.

Create a regex pattern to validate PIN code 400088



PIN Code UC 2

Restrict the PIN code from taking alphabets or special characters at the beginning.

Check for A400088 – this should fail



PIN Code UC 3

Restrict the PIN code from taking alphabets or special characters at the End.

Check for 400088B – this should fail



PIN Code UC 4

Make sure 400 088 is also
valid along with 400088



Email UC 1

Validate Email address with a regex. The email consists of minimum 3 and optional 2 more parts with mandatory @ and .

abc.xyz@bridgelabz.co.in

Here abc, bridgelabz and co are mandatory and the remaining 2 are optional

To begin with lets validate the mandatory part and start with abc



Email UC 2

Ensure @ and validate the mandatory 2nd part i.e. bridgelabz



Email UC 3

Ensure “.” after bridgelabz
and validate the
mandatory 3rd part i.e. co



Email UC 4

Lets handle optional part
i.e. xyz in

abc.xyz@bridgelabz.co.in

NOTE: make sure only
following are valid special
characters `_`, `+`, `-`, `.`

proceeding to xyz



Email UC 5

Finally lets close the expression with supporting optional parts.

Note: Top Level Domains (TLD) in the last part is the optional country code and its 2 characters only

Email Pattern and Execution Thread



```
#!/usr/local/bin/bash -x
```

```
echo "Enter Email Address "
```

```
read email
```

Part 1

Optional Part 2

Part 3

Part 4

```
emailPat="^[0-9a-zA-Z]+([._+-][0-9a-zA-Z]+)*@[0-9a-zA-Z]+.[a-zA-z]{2,4}([.][a-zA-z]{2})$"
```

Optional Part 5

```
if [[ $email =~ $emailPat ]]; then echo yes; else echo no; fi
```

```
pattern $ ./emailpattern.sh
```

```
+ echo 'Enter Email Address '
```

```
Enter Email Address
```

```
+ read email
```

```
abc.100@abc.com.au
```

```
+ emailPat='^[0-9a-zA-Z]+([._+-][0-9a-zA-Z]+)*@[0-9a-zA-Z]+.[a-zA-z]{2,4}([.][a-zA-z]{2})$'
```

```
+ [[ abc.100@abc.com.au =~ ^[0-9a-zA-Z]+([._+-][0-9a-zA-Z]+)*@[0-9a-zA-Z]+.[a-zA-z]{2,4}([.][a-zA-z]{2})$ ]]
```

```
+ echo yes
```

```
yes
```

Sample Emails to Test

• A. Valid Emails

- 1. abc@yahoo.com,
- 2. abc-100@yahoo.com,
- 3. abc.100@yahoo.com
- 2. abc111@abc.com,
- 4. abc-100@abc.net,
- 5. abc.100@abc.com.au
- 6. abc@1.com,
- 7. abc@gmail.com.com
- 8. abc+100@gmail.com

B. Invalid Emails (TLD - Top Level Domains)

1. abc – must contains “@” symbol
2. abc@.com.my – tld can not start with dot “.”
3. abc123@gmail.a – “.a” is not a valid tld, last tld must contains at least two characters
4. abc123@.com – tld can not start with dot “.”
5. abc123@.com.com – tld can not start with dot “.”
6. .abc@abc.com – email’s 1st character can not start with “.”
7. abc()*@gmail.com – email’s is only allow character, digit, underscore and dash
8. abc@%*.com – email’s tld is only allow character and digit
9. abc..2002@gmail.com – double dots “.” are not allow
10. abc.@gmail.com – email’s last character can not end with dot “.”
11. abc@abc@gmail.com – double “@” is not allow
12. abc@gmail.com.1a -email’s tld which has two characters can not contains digit
13. abc@gmail.com.aa.au - cannont have multiple email’s tld



BridgeLabz

Employability Delivered

Thank
You