Glob and Regex — Practice

CSCI 40

Fall 2026

Total Score: _____/4

LLM evaluation *2026-08-26*

Re-evaluation pending after quiz-bank corrections.

Note. Each snippet starts in a fresh, empty `quiz` directory. Setup commands are omitted.

The Glob and Regular Expressions

Note. In this notes packet, you will learn two new *query programming languages*: the Glob and Regular Expressions (regex). These languages are similar to CSS Selectors in that: (1) they are used to "find" information inside another filetype; (2) they are commonly used from within other languages.

1 Glob

Note. The glob is used for creating a "list" of files that match a particular pattern. You are responsible for the following syntax:

Pattern	Matches
`*`	any string (including empty)
`?`	exactly one character
`[abc]`	one character in the set
`[!abc]`	one character not in the set

1.1 With the Shell

Note. The glob is always applied to any arguments that are passed into a program.

Note. Anytime we write code that uses the glob, we say we are *globbing*.

Problem 1. Write the output of the final command in the following shell script.

```
$ echo 'hello world' > README.md
$ echo 'hola mundo' > README.txt
$ echo 'salve munde' > README
$ ls RE* | wc -l
```

Problem 2. Write the output of the final command in the following shell script.

```
$ echo 'hello world' > README.md
$ echo 'hola mundo' > README.txt
$ echo 'salve munde' > README
$ ls *.txt | wc -l
```

Problem 3. Write the output of the final command in the following shell script.

```
$ echo 'hello world' > README.md
$ echo 'hola mundo' > README.txt
$ echo 'salve munde' > README
$ cat *.* | wc -l
```

Problem 4. Write the output of the final command in the following shell script.

```
$ echo 'hello world'      > README.en # english
$ echo 'hola mundo'       > README.es # spanish
$ echo 'salve munde'      > README.la # latin
$ echo 'hallo welt'       > README.de # german
$ echo 'ola mundo'        > README.pt # portuguese
$ echo 'bonjour le monde' > README.fr # french
$ echo 'saluton mondo'    > README.epo # esperanto
$ cat *e | wc -l
```

Problem 5. Write the output of the final command in the following shell script.

```
$ echo 'hello world'      > README.en # english
$ echo 'hola mundo'       > README.es # spanish
$ echo 'salve munde'      > README.la # latin
$ echo 'hallo welt'       > README.de # german
$ echo 'ola mundo'        > README.pt # portuguese
$ echo 'bonjour le monde' > README.fr # french
$ echo 'saluton mondo'    > README.epo # esperanto
$ cat README.?? | wc -l
```

Problem 6. Write the output of the final command in the following shell script.

```
$ echo 'hello world'      > README.en # english
$ echo 'hola mundo'       > README.es # spanish
$ echo 'salve munde'      > README.la # latin
$ echo 'hallo welt'       > README.de # german
$ echo 'ola mundo'        > README.pt # portuguese
$ echo 'bonjour le monde' > README.fr # french
$ echo 'saluton mondo'    > README.epo # esperanto
$ cat README.??? | wc -l
```

Problem 7. Write the output of the final command in the following shell script.

```
$ echo 'hello world'      > README.en # english
$ echo 'hola mundo'       > README.es # spanish
$ echo 'salve munde'      > README.la # latin
$ echo 'hallo welt'       > README.de # german
$ echo 'ola mundo'        > README.pt # portuguese
$ echo 'bonjour le monde' > README.fr # french
$ echo 'saluton mondo'    > README.epo # esperanto
$ cat README.e? | wc -l
```

Problem 8. Write the output of the final command in the following shell script.

```
$ echo 'hello world'      > README.en # english
$ echo 'hola mundo'       > README.es # spanish
$ echo 'salve munde'      > README.la # latin
$ echo 'hallo welt'       > README.de # german
$ echo 'ola mundo'        > README.pt # portuguese
$ echo 'bonjour le monde' > README.fr # french
$ echo 'saluton mondo'    > README.epo # esperanto
$ cat README.[elf]? | wc -l
```

Problem 9. Write the output of the final command in the following shell script.

```
$ echo 'hello world'      > README.en # english
$ echo 'hola mundo'       > README.es # spanish
$ echo 'salve munde'      > README.la # latin
$ echo 'hallo welt'       > README.de # german
$ echo 'ola mundo'        > README.pt # portuguese
$ echo 'bonjour le monde' > README.fr # french
$ echo 'saluton mondo'    > README.epo # esperanto
$ cat README.[!elf]? | wc -l
```

Problem 10. Write the output of the final command in the following shell script.

```
$ mkdir romance
$ mkdir germanic
$ echo 'hello world'      > romance/README.en # english
$ echo 'hola mundo'       > romance/README.es # spanish
$ echo 'salve munde'      > romance/README.la # latin
$ echo 'hallo welt'       > germanic/README.de # german
$ echo 'hallo wereld'     > germanic/README.nl # dutch
$ echo 'hej verden'       > germanic/README.da # danish
$ echo 'hei verden'       > germanic/README.no # norwegian
$ echo 'ola mundo'        > romance/README.pt # portuguese
$ echo 'bonjour le monde' > romance/README.fr # french
$ echo 'saluton mondo'    > romance/README.epo # esperanto
$ cat romance/README.[!elf]? | wc -l
```

Problem 11. Write the output of the final command in the following shell script.

```
$ mkdir romance
$ mkdir germanic
$ echo 'hello world'      > romance/README.en # english
$ echo 'hola mundo'       > romance/README.es # spanish
$ echo 'salve munde'      > romance/README.la # latin
$ echo 'hallo welt'       > germanic/README.de # german
$ echo 'hallo wereld'     > germanic/README.nl # dutch
$ echo 'hej verden'       > germanic/README.da # danish
$ echo 'hei verden'       > germanic/README.no # norwegian
$ echo 'ola mundo'        > romance/README.pt # portuguese
$ echo 'bonjour le monde' > romance/README.fr # french
$ echo 'saluton mondo'    > romance/README.epo # esperanto
$ cat */README.[!elf]? | wc -l
```

Problem 12. Write the output of the final command in the following shell script.

```
$ mkdir romance
$ mkdir germanic
$ echo 'hello world'      > romance/README.en # english
$ echo 'hola mundo'       > romance/README.es # spanish
$ echo 'salve munde'      > romance/README.la # latin
$ echo 'hallo welt'       > germanic/README.de # german
$ echo 'hallo wereld'     > germanic/README.nl # dutch
$ echo 'hej verden'       > germanic/README.da # danish
$ echo 'hei verden'       > germanic/README.no # norwegian
$ echo 'ola mundo'        > romance/README.pt # portuguese
$ echo 'bonjour le monde' > romance/README.fr # french
$ echo 'saluton mondo'    > romance/README.epo # esperanto
$ cat *a*/README.??? | wc -l
```

1.2 With Python

Note. Python has a built-in module `glob` for globbing. The function `glob.glob` returns a list of all paths that match the glob expression.

Note. The problems below are identical to problems in the previous section, except that a python file `example.py` is created and the glob calculated inside this file. DO NOT FORGET THAT THE `example.py` FILE MIGHT MATCH THE GLOB PATTERNS!

Problem 13. Write the output of the final command in the following shell script.

```
$ echo 'hello world' > README.md
$ echo 'hola mundo' > README.txt
$ echo 'salve munde' > README
$ cat > example.py <<EOF
import glob
paths = glob.glob('*.*')
print(len(paths))
EOF
$ python3 example.py
```

Problem 14. Write the output of the final command in the following shell script.

```
$ echo 'hello world'      > README.en # english
$ echo 'hola mundo'       > README.es # spanish
$ echo 'salve munde'      > README.la # latin
$ echo 'hallo welt'       > README.de # german
$ echo 'ola mundo'        > README.pt # portuguese
$ echo 'bonjour le monde' > README.fr # french
$ echo 'saluton mondo'    > README.epo # esperanto
$ cat > example.py <<EOF
import glob
paths = glob.glob('README.[elf]?')
print(len(paths))
EOF
$ python3 example.py
```

Problem 15. Write the output of the final command in the following shell script.

```
$ mkdir romance
$ mkdir germanic
$ echo 'hello world'      > romance/README.en # english
$ echo 'hola mundo'       > romance/README.es # spanish
$ echo 'salve munde'      > romance/README.la # latin
$ echo 'hallo welt'       > germanic/README.de # german
$ echo 'hallo wereld'     > germanic/README.nl # dutch
$ echo 'hej verden'       > germanic/README.da # danish
$ echo 'hei verden'       > germanic/README.no # norwegian
$ echo 'ola mundo'        > romance/README.pt # portuguese
$ echo 'bonjour le monde' > romance/README.fr # french
$ echo 'saluton mondo'    > romance/README.epo # esperanto
$ cat > example.py <<EOF
import glob
paths = glob.glob('*/README.[!elf]?')
print(len(paths))
EOF
$ python3 example.py
```

Note. Recall that we are not required to use the `glob` module within python in order to match filenames. This happens at the shell-level, and so all programs that accept arguments get this ability.

Problem 16. Write the output of the final command in the following shell script.

```
$ echo 'hello world' > README.md
$ echo 'hola mundo' > README.txt
$ echo 'salve munde' > README
$ cat > example.py <<EOF
import sys
print(len(sys.argv))
EOF
$ python3 example.py *.*
```

Problem 17. Write the output of the final command in the following shell script.

```
$ echo 'hello world'      > README.en # english
$ echo 'hola mundo'       > README.es # spanish
$ echo 'salve munde'      > README.la # latin
$ echo 'hallo welt'       > README.de # german
$ echo 'ola mundo'        > README.pt # portuguese
$ echo 'bonjour le monde' > README.fr # french
$ echo 'saluton mondo'    > README.epo # esperanto
$ cat > example.py <<EOF
import sys
print(len(sys.argv))
EOF
$ python3 example.py README.[elf]?
```

Problem 18. Write the output of the final command in the following shell script.

```
$ mkdir romance
$ mkdir germanic
$ echo 'hello world'      > romance/README.en # english
$ echo 'hola mundo'       > romance/README.es # spanish
$ echo 'salve munde'      > romance/README.la # latin
$ echo 'hallo welt'       > germanic/README.de # german
$ echo 'hallo wereld'     > germanic/README.nl # dutch
$ echo 'hej verden'       > germanic/README.da # danish
$ echo 'hei verden'       > germanic/README.no # norwegian
$ echo 'ola mundo'        > romance/README.pt # portuguese
$ echo 'bonjour le monde' > romance/README.fr # french
$ echo 'saluton mondo'    > romance/README.epo # esperanto
$ cat > example.py <<EOF
import sys
print(len(sys.argv))
EOF
$ python3 example.py */README.[!elf]?
```

2 Regex

Note. Regular expressions (regex) are closely related to the glob:

1. The glob is generally used for finding files based on their *name*, and regex is used for finding files based on their *contents*.
2. Regex has a syntax similar to the glob, but the semantics is slightly different. It is easy to get them confused, so pay attention!

You are responsible for the following regex syntax:

Pattern	Matches
`.`	exactly one character
`*`	zero or more of preceding element
`+`	one or more of preceding element
`?`	zero or one of preceding element
`[abc]`	one character in the set
`[^abc]`	one character not in the set
`^`	start of string
`$`	end of string
`(a|b)`	alternation (a or b)

Note. For space reasons, I am giving you a very limited set of regex/glob patterns below. You should try creating your own problems with your own regex/glob patterns to check your understanding. This is another example of me trying to get you to learn-how-to-learn.

2.1 In the shell

Note. The shell program for using regex is called `grep`. It is so commonly used that programmers often say we are "grepping" for something whenever we are "searching" for something (similar to how muggles will often say they are "googling" something even if they are using another search engine like Bing, Baidu, or Yandex).

Problem 19. Write the output of the final command in the following shell script.

```
$ mkdir romance
$ mkdir germanic
$ echo 'hello world'      > romance/README.en # english
$ echo 'hola mundo'       > romance/README.es # spanish
$ echo 'salve munde'      > romance/README.la # latin
$ echo 'hallo welt'       > germanic/README.de # german
$ echo 'hallo wereld'     > germanic/README.nl # dutch
$ echo 'hej verden'       > germanic/README.da # danish
$ echo 'hei verden'       > germanic/README.no # norwegian
$ echo 'ola mundo'        > romance/README.pt # portuguese
$ echo 'bonjour le monde' > romance/README.fr # french
$ echo 'saluton mondo'    > romance/README.epo # esperanto
$ cat romance/* | grep 'o' | wc -l
```

Problem 20. Write the output of the final command in the following shell script.

```
$ mkdir romance
$ mkdir germanic
$ echo 'hello world'      > romance/README.en # english
$ echo 'hola mundo'       > romance/README.es # spanish
$ echo 'salve munde'      > romance/README.la # latin
$ echo 'hallo welt'       > germanic/README.de # german
$ echo 'hallo wereld'     > germanic/README.nl # dutch
$ echo 'hej verden'       > germanic/README.da # danish
$ echo 'hei verden'       > germanic/README.no # norwegian
$ echo 'ola mundo'        > romance/README.pt # portuguese
$ echo 'bonjour le monde' > romance/README.fr # french
$ echo 'saluton mondo'    > romance/README.epo # esperanto
$ cat romance/* | grep '^o' | wc -l
```

Problem 21. Write the output of the final command in the following shell script.

```
$ mkdir romance
$ mkdir germanic
$ echo 'hello world'      > romance/README.en # english
$ echo 'hola mundo'       > romance/README.es # spanish
$ echo 'salve munde'      > romance/README.la # latin
$ echo 'hallo welt'       > germanic/README.de # german
$ echo 'hallo wereld'     > germanic/README.nl # dutch
$ echo 'hej verden'       > germanic/README.da # danish
$ echo 'hei verden'       > germanic/README.no # norwegian
$ echo 'ola mundo'        > romance/README.pt # portuguese
$ echo 'bonjour le monde' > romance/README.fr # french
$ echo 'saluton mondo'    > romance/README.epo # esperanto
$ cat romance/* | grep 'o$' | wc -l
```

Problem 22. Write the output of the final command in the following shell script.

```
$ mkdir romance
$ mkdir germanic
$ echo 'hello world'      > romance/README.en # english
$ echo 'hola mundo'       > romance/README.es # spanish
$ echo 'salve munde'      > romance/README.la # latin
$ echo 'hallo welt'       > germanic/README.de # german
$ echo 'hallo wereld'     > germanic/README.nl # dutch
$ echo 'hej verden'       > germanic/README.da # danish
$ echo 'hei verden'       > germanic/README.no # norwegian
$ echo 'ola mundo'        > romance/README.pt # portuguese
$ echo 'bonjour le monde' > romance/README.fr # french
$ echo 'saluton mondo'    > romance/README.epo # esperanto
$ cat romance/* | grep 'o.*d' | wc -l
```

Problem 23. Write the output of the final command in the following shell script.

```
$ mkdir romance
$ mkdir germanic
$ echo 'hello world'      > romance/README.en # english
$ echo 'hola mundo'       > romance/README.es # spanish
$ echo 'salve munde'      > romance/README.la # latin
$ echo 'hallo welt'       > germanic/README.de # german
$ echo 'hallo wereld'     > germanic/README.nl # dutch
$ echo 'hej verden'       > germanic/README.da # danish
$ echo 'hei verden'       > germanic/README.no # norwegian
$ echo 'ola mundo'        > romance/README.pt # portuguese
$ echo 'bonjour le monde' > romance/README.fr # french
$ echo 'saluton mondo'    > romance/README.epo # esperanto
$ cat romance/* | grep -E 'hello|salve|hallo' | wc -l
```

2.2 Within Python

Note. Python has a built-in module `re` for working with regex. The function `re.search` returns a match object if the input `str` matches the regex, and it returns `None` otherwise. A match object is truthy and `None` is falsy, so the result works directly in an `if` condition. It is customary (but not required) in python to use *raw-strings* to define regex. (Recall that raw string literals are prefixed with the letter `r` like `r'...'` and that the backslash escape characters like `\n` are not interpreted inside of rawstrings.)

Problem 24. Write the output of the final command in the following shell script.

```
$ mkdir romance
$ mkdir germanic
$ echo 'hello world'      > romance/README.en # english
$ echo 'hola mundo'       > romance/README.es # spanish
$ echo 'salve munde'      > romance/README.la # latin
$ echo 'hallo welt'       > germanic/README.de # german
$ echo 'hallo wereld'     > germanic/README.nl # dutch
$ echo 'hej verden'       > germanic/README.da # danish
$ echo 'hei verden'       > germanic/README.no # norwegian
$ echo 'ola mundo'        > romance/README.pt # portuguese
$ echo 'bonjour le monde' > romance/README.fr # french
$ echo 'saluton mondo'    > romance/README.epo # esperanto
$ cat > example.py <<EOF
import glob
import re
count = 0
for filepath in glob.glob('romance/*'):
    with open(filepath) as f:
        for line in f:
            if re.search(r'o.*d', line):
                count += 1
print(count)
EOF
$ python example.py
```

Problem 25. Write the output of the final command in the following shell script.

```
$ mkdir romance
$ mkdir germanic
$ echo 'hello world'      > romance/README.en # english
$ echo 'hola mundo'       > romance/README.es # spanish
$ echo 'salve munde'      > romance/README.la # latin
$ echo 'hallo welt'       > germanic/README.de # german
$ echo 'hallo wereld'     > germanic/README.nl # dutch
$ echo 'hej verden'       > germanic/README.da # danish
$ echo 'hei verden'       > germanic/README.no # norwegian
$ echo 'ola mundo'        > romance/README.pt # portuguese
$ echo 'bonjour le monde' > romance/README.fr # french
$ echo 'saluton mondo'    > romance/README.epo # esperanto
$ cat > example.py <<EOF
import glob
import re
count = 0
for filepath in glob.glob('germanic/*'):
    with open(filepath) as f:
        for line in f:
            if re.search(r'd$', line):
                count += 1
print(count)
EOF
$ python example.py
```

Problem 26. Write the output of the final command in the following shell script.

```
$ mkdir romance
$ mkdir germanic
$ echo 'hello world'      > romance/README.en # english
$ echo 'hola mundo'       > romance/README.es # spanish
$ echo 'salve munde'      > romance/README.la # latin
$ echo 'hallo welt'       > germanic/README.de # german
$ echo 'hallo wereld'     > germanic/README.nl # dutch
$ echo 'hej verden'       > germanic/README.da # danish
$ echo 'hei verden'       > germanic/README.no # norwegian
$ echo 'ola mundo'        > romance/README.pt # portuguese
$ echo 'bonjour le monde' > romance/README.fr # french
$ echo 'saluton mondo'    > romance/README.epo # esperanto
$ cat > example.py <<EOF
import glob
import re
count = 0
for filepath in glob.glob('*/*'):
    with open(filepath) as f:
        for line in f:
            if re.search(r'salve|hallo|hello', line):
                count += 1
print(count)
EOF
$ python example.py
```