ANSWERS

A6.4 Regular expressions and regular languages

Theory of computation · A level · AQA 7517 4.4.2.3 · about 25 min

What this lesson is about

The metacharacters, matching in Python, the link between regexes and FSMs, and what makes a language regular.

- Languages
- The metacharacters
- Regular expressions in Python
- Regular expressions and FSMs
- Regular languages

Questions

6 marks in all

1. Which strings match the regular expression ab*a? [1 mark]

Tick every answer that is true.

- ☐ **A** aa
- ☐ **B** aba
- ☐ **C** abba
- ☐ **D** ab
- ☐ **E** baa

> **Answer:** A, B, C. ab*a is an a, zero or more b, then an a. ab has no final a, and baa starts with b.

2. What does the regular expression (a|b)+ describe? [1 mark]

- ☐ **A** Every string of a and b, including the empty string
- ☐ **B** Every non-empty string of a and b
- ☐ **C** Exactly one a or one b
- ☐ **D** The string ab repeated

> **Answer:** B. + means one or more, so the empty string is excluded. (a|b)* would include it.

3. Which statement about regular expressions and finite state machines is true? [1 mark]

- ☐ **A** Every regex has an FSM that accepts the same language
- ☐ **B** FSMs can describe more languages than regexes
- ☐ **C** Regexes can describe more languages than FSMs
- ☐ **D** Only regexes can describe infinite languages

> **Answer:** A. They are equivalent: any language a regex describes, some FSM accepts, and the other way round.

4. Why is the language $\{0^n1^n \mid n \geq 1\}$ not regular? [1 mark]

- ☐ **A** It is infinite
- ☐ **B** An FSM would need to count an unlimited number of 0s with a finite number of states
- ☐ **C** It uses two symbols
- ☐ **D** It contains the empty string

> **Answer:** B. Many regular languages are infinite. This one needs unlimited counting, which a finite set of states cannot do.

5. Write a regular expression, using only 0, 1 and the metacharacters * + ? | ( ), for binary strings that start with 1 and end with 0. [1 mark]

> **Answer:** 1(0|1)*0. Fix the first and last symbols, and allow any binary digits in between with (0|1)*.

6. What does this program print? [1 mark]

```
import re
for s in ["ac", "abbc", "abcc", "c"]:
    print(s, re.fullmatch(r"ab*c", s) is not None)
```

> **Answer:**
>
> ```
> ac True
> abbc True
> abcc False
> c False
> ```
>
> ab*c needs an a, any number of b, then exactly one c, and fullmatch needs the whole string to fit.

The task: three languages

Write a regular expression for each language, then test it. - `L1` : strings over `{a, b}` that start with `a` and end with `b` . - `L2` : binary strings in which every `1` is immediately followed by a `0` . The string `000` counts, because it has no 1s at all. - `L3` : binary numbers with no leading zeros: `0` itself, or a `1` followed by any binary digits. Rules: - Assign each pattern to its variable as a raw string on one line, such as `L1 = r"..."` . Use only the symbols of the language and the metacharacters `* + ? | ( )` . No `[ ]` , no `\d` . - Test with `re.fullmatch` . For every string in `tests[name]` , working through `L1` , `L2` , `L3` in that order, print the name, the string and `yes` or `no` , separated by single spaces: for example `L1 ab yes` . Fifteen lines in all.

```
# the two lines every program starts with: the commands, then the robot
from bugbot import *
connect()

import re

tests = {
    "L1": ["ab", "aab", "ba", "a", "abab"],
    "L2": ["1010", "0100", "110", "01", "000"],
    "L3": ["0", "1010", "0101", "100", "00"],
}
```

The hint students can ask for: For L1, fix the first and last symbols and let anything go between. For L2, think of the string as built from blocks, where a block is either a lone 0 or a 1 with its 0. For L3, there are two kinds of valid string, so use alternation. Test each pattern against the strings that should fail as well as those that should match.

A solution

```
# the two lines every program starts with: the commands, then the robot
from bugbot import *
connect()

import re

tests = {
    "L1": ["ab", "aab", "ba", "a", "abab"],
    "L2": ["1010", "0100", "110", "01", "000"],
    "L3": ["0", "1010", "0101", "100", "00"],
}
L1 = r"a(a|b)*b"
L2 = r"(0|10)*"
L3 = r"0|1(0|1)*"
patterns = {"L1": L1, "L2": L2, "L3": L3}
for name in ["L1", "L2", "L3"]:
    for text in tests[name]:
        print(name, text, "yes" if re.fullmatch(patterns[name], text) else "no")
```

Any program that meets the task's checks is marked correct in the simulator; this is one way, not the only way.