
BREAKING THE ICE
WITH
REGULAR EXPRESSIONS

Contents

Joining the Voyage

Find each shipmate's answer as to whether they are willing to voyage!

Subject Strings	Accepted Answer Keywords
ok, i will do it	ok
okie dokie	Okay
Ahooy, Okay!!!	sure
why sure, i can go	yes
arrr, yes matey	y
My answer me mate, is yes	
y	

Starting the Expression by Matching the First Confirmation

using ok literal to directly match "ok"

Regular Expression

```
/ok/
```

Possible Subjects

ok, i will do it

okie dokie

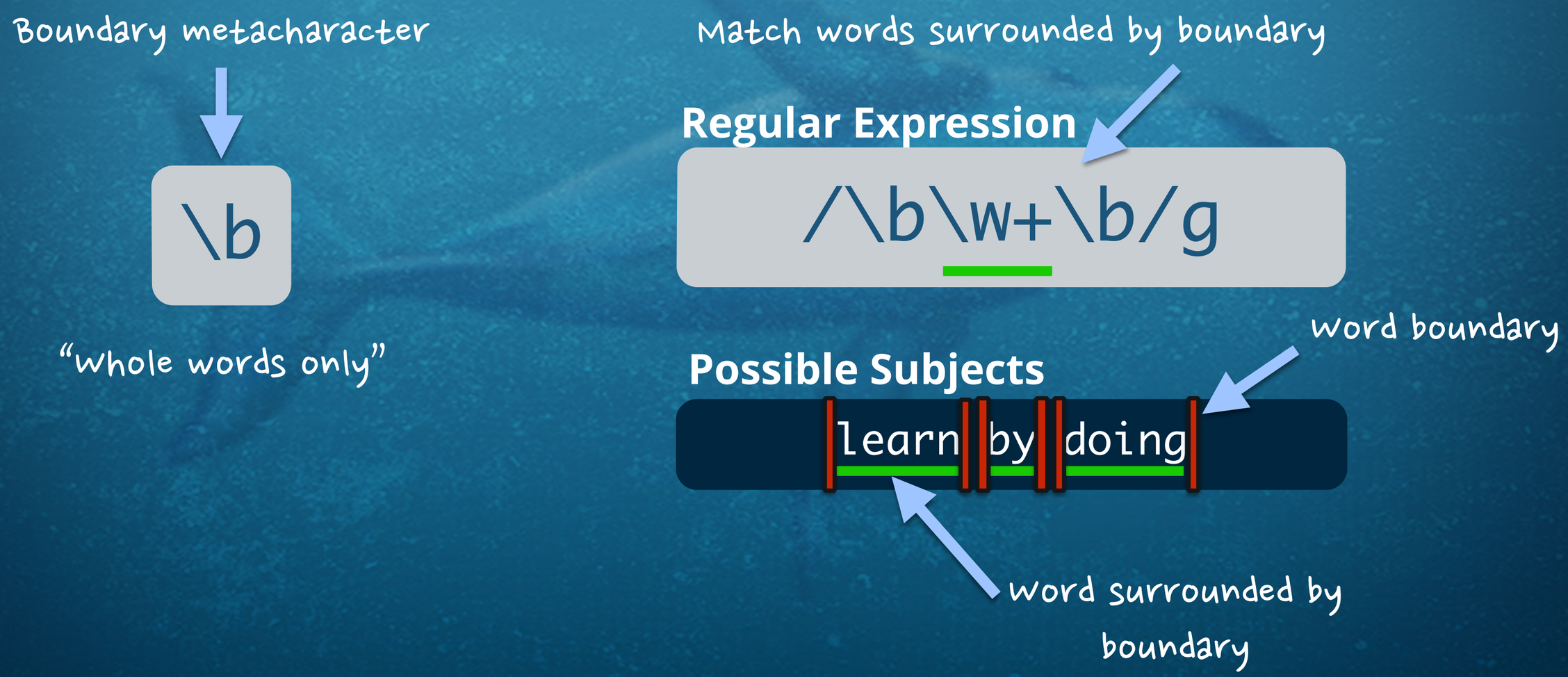
Pattern Goals

- ok
- Okay
- sure
- yes
- y
- answer by itself
- ignore other text

we want the answer to be by itself and not part of another word

BREAKING THE ICE WITH REGULAR EXPRESSIONS

Detecting Word Boundaries With the Boundary Metacharacter



Using Boundaries in a Pattern

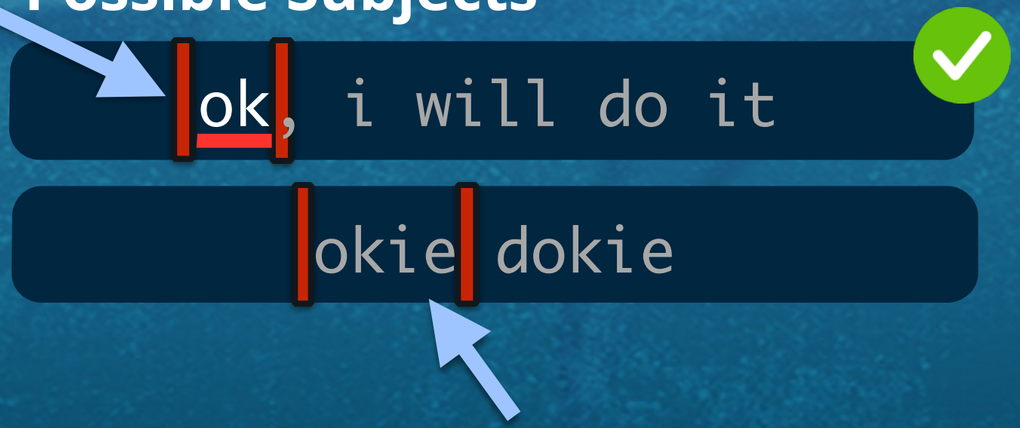
Regular Expression

```
/\bok\b/
```

Ensures our match is a single word

`\b` = boundary metacharacter

Possible Subjects



Great - we don't get a match because "ok" does not have a boundary on the right, it only has letters

Pattern Goals

- ✅ ok
  - Okay
  - sure
  - yes
  - y
- ✅ answer by itself
- ✅ ignore other text

Problem: We Need to Match Multiple Versions of a Word

Regular Expression

```
/\bok\b/
```

Possible Subjects

```
Ahooy, Okay!!!
```

wait - we want "okay" to match too

Pattern Goals

- ✅ ok
  - Okay
  - sure
  - yes
  - y
- ✅ answer by itself
- ✅ ignore other text

Inefficient Matching With Multiple OR Statements

Regular Expression

```
/\bok\b|\bokay\b/
```

"ok" OR "okay"

But what if there were 50 different options?

If we had a way to make "ay" optional, we could match with a single pattern.

```
/\bok(ay)\b/
```

How can we make this optional?

Pattern Goals

- ✅ ok
  Okay
  sure
  yes
  y
- ✅ answer by itself
- ✅ ignore other text

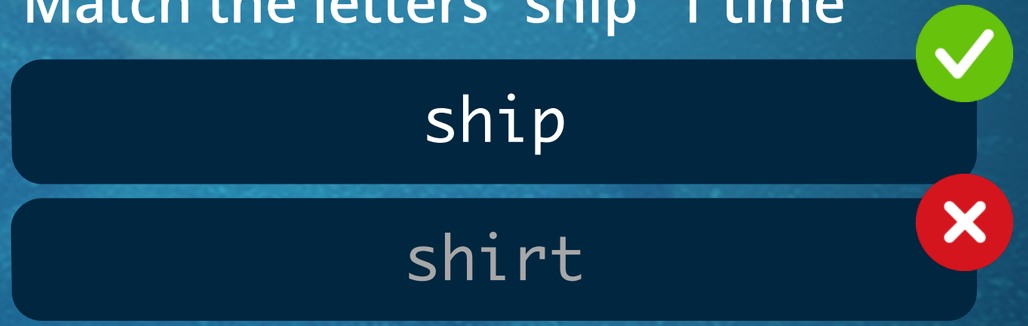
Checking for a Pattern 0 or More Times

Regular Expression

```
/ship/
```

All of these characters are required

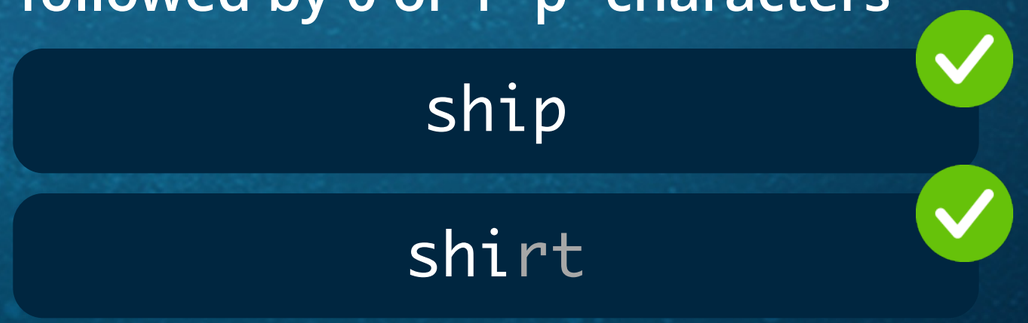
Match the letters “ship” 1 time




```
/ship?/
```

characters in a pattern that are followed by a question mark are optional

Match the letters “shi” 1 time followed by 0 or 1 “p” characters



Marking a Group of Characters as Optional

Regular Expression

```
/pirate\s(ship)?/
```

Group characters in parentheses followed by "?" to make them all optional

```
pirate ship
```

```
pirate boat
```

Matching Different Versions of the Same Word

Regular Expression

Match 0 or 1 times

"i" modifier for upper- and lowercase

```
/\bok(ay)?\b/i
```

Possible Subjects

Ahooy, Okay!!!

ok, i will do it

Pattern Goals

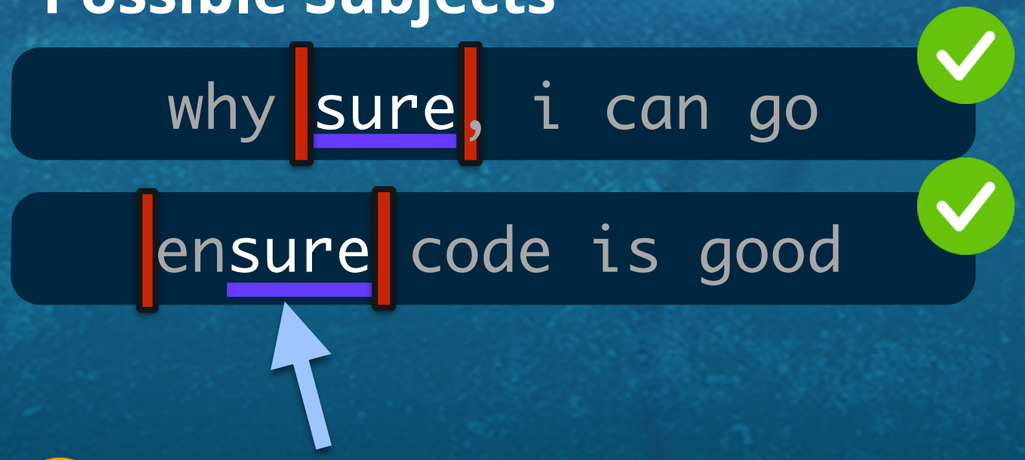
- ok
- Okay
- sure
- yes
- y
- answer by itself
- ignore other text

Problem: The OR Operator Is Splitting the Boundary

Regular Expression

```
/\bok(ay)?|sure\b/i
```

Possible Subjects



Pattern Goals

- ✓ ok
- ✓ Okay
- ✓ sure
- yes
- y
- ✗ answer by itself
- ✓ ignore other text

! we do not want a partial match of the answer

Solution: Group the Pattern Within the Boundary

Group ensures boundary is applied to all answers

Regular Expression

```
/\b(ok(ay)?|sure)\b/i
```

Pattern Goals

- ok
- Okay
- sure
- yes
- y
- answer by itself
- ignore other text

Possible Subjects

why sure, i can go

ensure code is good

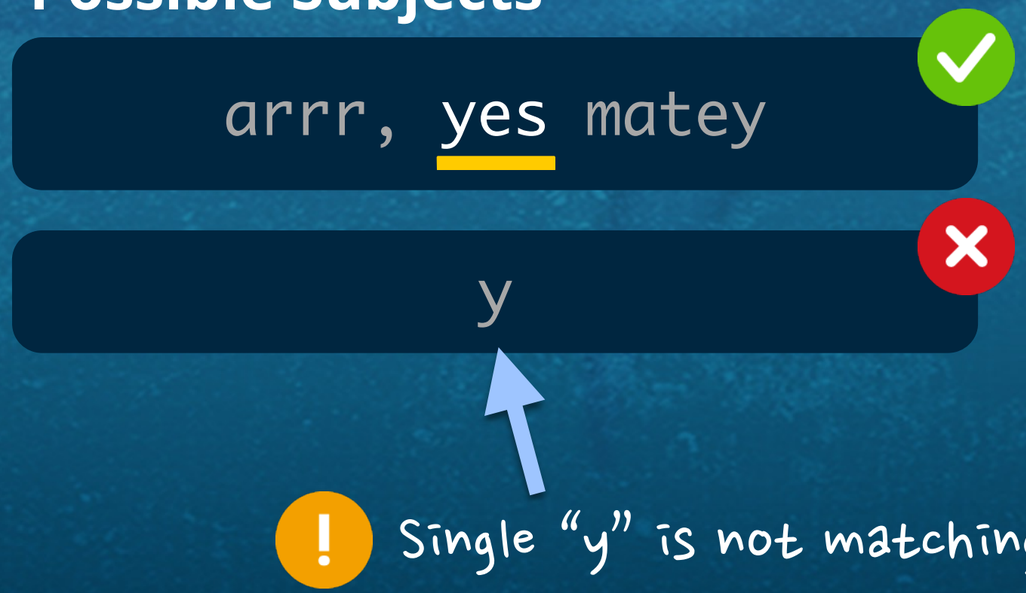
Great - "ensure" is no longer matched

Continuing to Match More Valid Subjects

Regular Expression

```
/\b(ok(ay)?|sure|yes)\b/i
```

Possible Subjects



Pattern Goals

- ok
- Okay
- sure
- yes

  y
- answer by itself
- ignore other text

Another Optional Part of the Answer

Regular Expression

```
/\b(ok(ay)?|surely(es)?)\b/i
```

Possible Subjects

arrr, yes matey

y

Pattern Goals

- ok
- Okay
- sure
- yes
- y
- answer by itself
- ignore other text

Our Finished Pattern

Regular Expression

```
/\b(ok(ay)?|sure|y(es)?)\b/i
```

Possible Subjects

- ok, i will do it
- Ahooy, Okay!!!
- why sure, i can go
- arr, yes matey
- y

Pattern Goals

- ok
- Okay
- sure
- yes
- y
- answer by itself
- ignore other text

Section 2

Confirmative

Sailor Taglines

All sailors have entered their taglines!

Work like a captain, play like a pirate.

Keep calm and say Arr.

Shiver me timbers matey.

Why are pirates pirates? cuz they arr.

Pattern Goals

does not contain numbers

40 characters or less

20 characters or more

Start With a Base Pattern for Matching the Tagline

Regular Expression

```
/[a-z]+/i
```

Subject

```
Work like a captain, play
like a pirate.
```

Not matching whitespace

Pattern Goals

- does not contain numbers
- 40 characters or less
- 20 characters or more

Expanding the Pattern to Include Whitespace

Regular Expression

```
/[a-z\s]+/i
```

Subject

```
Work like a captain, play
like a pirate.
```

we also need to match the comma

Pattern Goals

- does not contain numbers
- 40 characters or less
- 20 characters or more

Including A Comma

Regular Expression

```
/[a-z\s,]+/i
```

Subject

```
Work like a captain, play like a pirate.
```

Pattern Goals

- does not contain numbers
- 40 characters or less
- 20 characters or more

Writing a Shorter Pattern With the NOT Operator

Everything matched with this pattern…

```
/[a-z\s,]+/i
```

…is also matched by this one.

```
/[^\d]+/i
```

The "^" means "not" when placed within a character set

This means "any number"

Subject

Work like a captain, play like a pirate.

so, this pattern means "anything that's not a number"

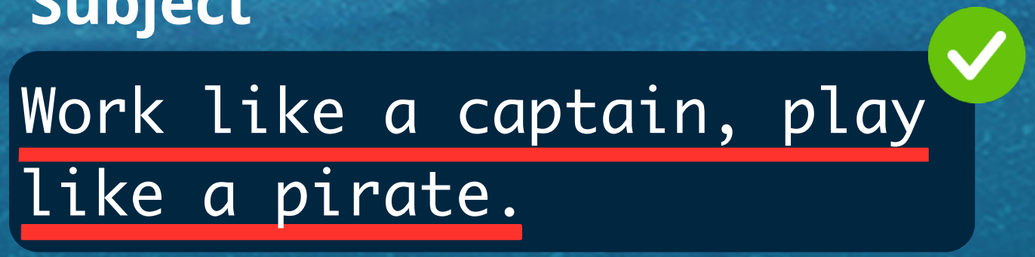
1 Caret Symbol — 2 Different Meanings

^ used to anchor beginning of subject

^ used to negate proceeding pattern

Regular Expression

Subject



Pattern Goals

- ✅ does not contain numbers
- 40 characters or less
- 20 characters or more

`\d` = digit character (number)

`^` = not the following character(s)

Even Shorter With Negated Shorthand Characters

`[^\d]` is the same as `\D`

match every character except numbers

`[^\s]` is the same as `\S`

match every character except whitespace

`[^\w]` is the same as `\W`

match every character except words

NOTE: You don't need a character set when using negated shorthand characters

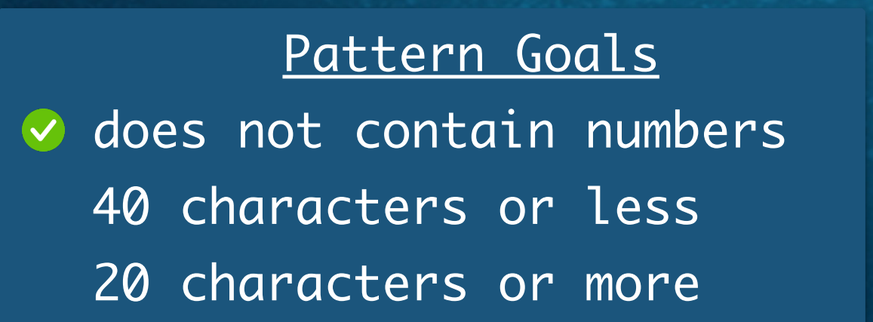
Using the Negated Shorthand Syntax in a Pattern

Regular Expression

```
/^\D+$/
```

Subject

```
Work like a captain, play like a pirate.
```



Problem: We Want to Set Min and Max Characters

Regular Expression

```
/^\D+$/
```

we want a minimum of 20 characters...

Subject

Work like a

Subject

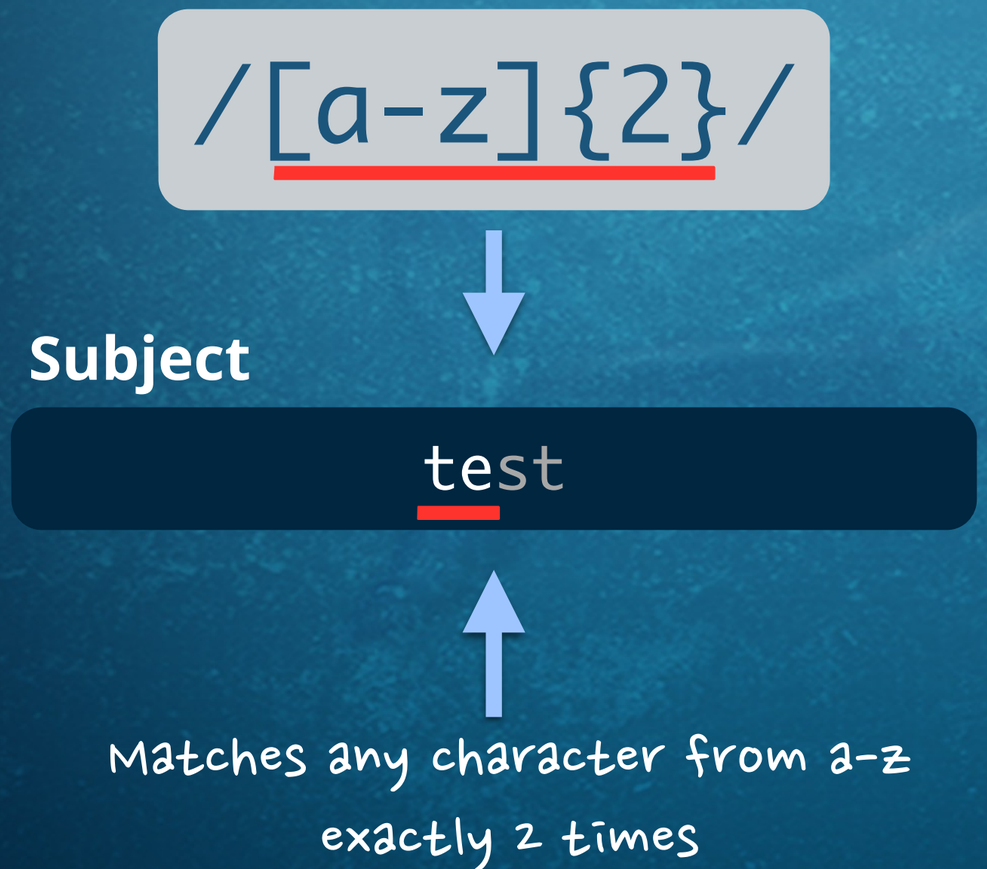
Work like a captain, play like a pirate. Work like a captain, play like a pirate. Work like a captain, play like a pirate.

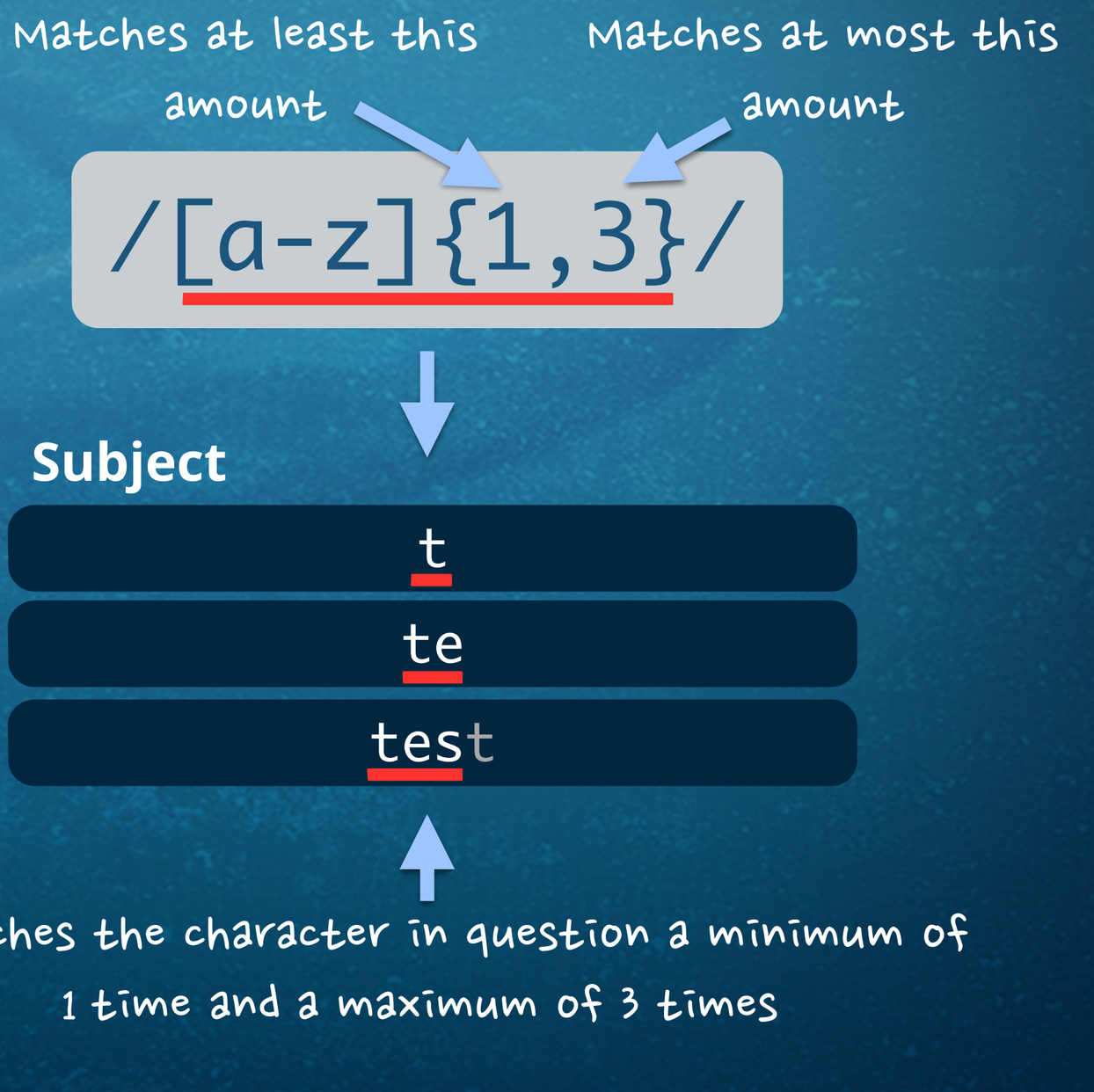
...and a maximum of 40 characters

Pattern Goals

- does not contain numbers
- 40 characters or less
- 20 characters or more

Matching a Specific Number of Times With Interval Expressions





First, Set the Minimum Amount of Characters

Ensure length is greater than or equal to 20

Regular Expression

```
/\D{20,}/i
```

Subject with at least 20 characters is matched

Subject

```
Work like a captain, play like a pirate.
```

Subject

```
Work like a captain.
```

Great! Subject with length less than 20 characters is not matched.

Pattern Goals

- does not contain numbers
- 40 characters or less
- 20 characters or more

Next, Set a Maximum Number of Characters

Regular Expression

```
/^\D{20,40}$/
```

Subject between 20 and 40 characters matched

Subject

Work like a captain, play like a pirate.

Great! Long subject no longer matched.

Work like a captain, play like a pirate. Work like a captain, play like a pirate. Work like a captain, play like a pirate.

Pattern Goals

- does not contain numbers
- 40 characters or less
- 20 characters or more