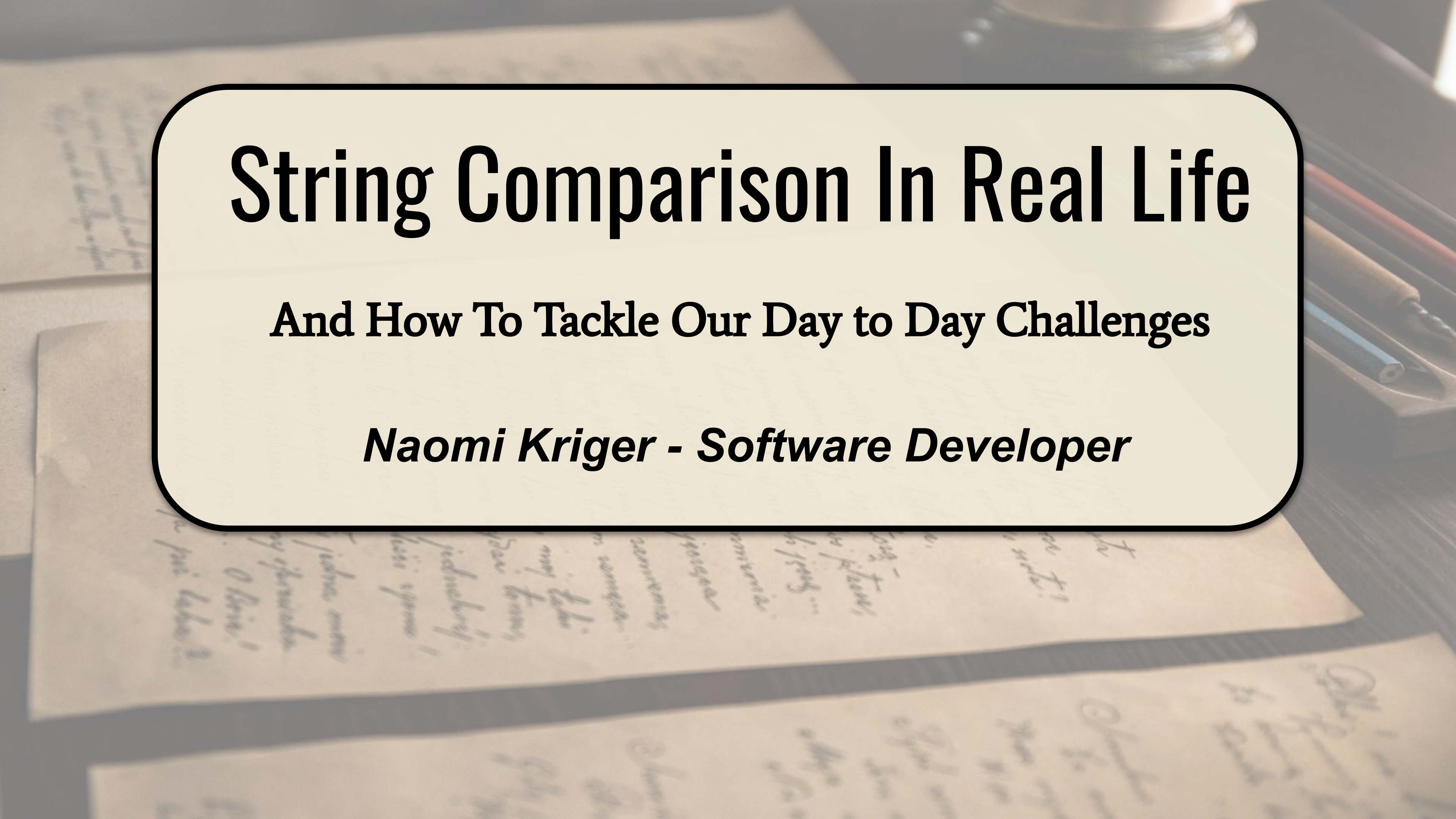


String Comparison In Real Life

And How To Tackle Our Day to Day Challenges

Naomi Kriger - Software Developer

Table A

Hey hi	sin
name	me

Table B

hey	sing
name!	ME!

Table A

Hey hi	sin
name	me

Table B

hey	sing
name!	ME!

Similarity(“name”, “name!”) = 9/10

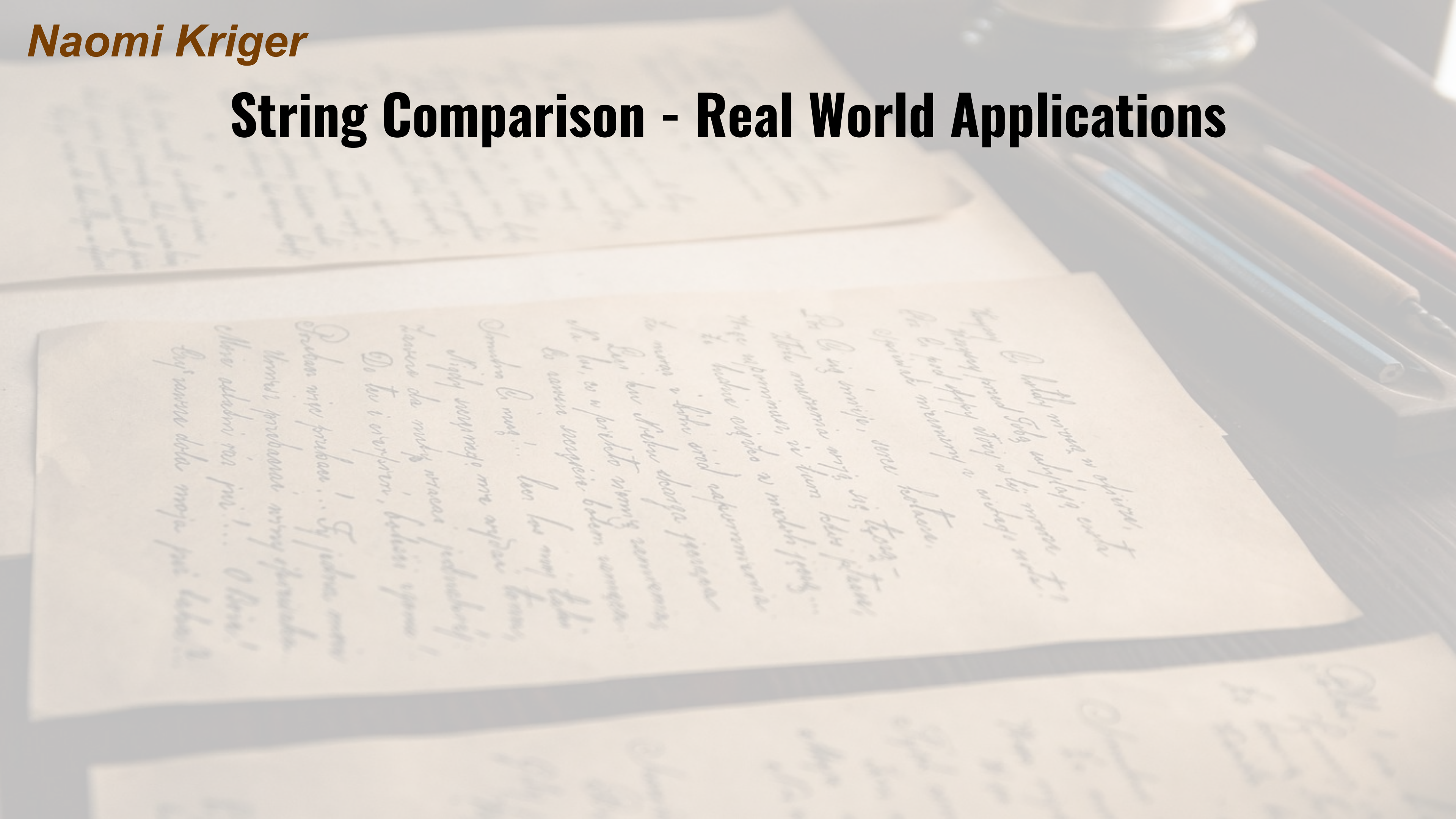
String Comparison - What Could Go Wrong?

- Typos
- Abbreviations
- Reordering of words in a sentence
- Repetitions of words
- Punctuation

There is no single definition for similarity or difference of strings

Naomi Kriger

String Comparison - Real World Applications



String Comparison - Real World Applications

- Fraud detection

String Comparison - Real World Applications

- Fraud detection

Identities Commonly Used By Fraudsters

- George Forge
- Billy Stealy
- Jerry Robbery
- ...

Naomi Kriger

String Comparison - Real World Applications

- Fraud detection

ORIGINAL PERSON

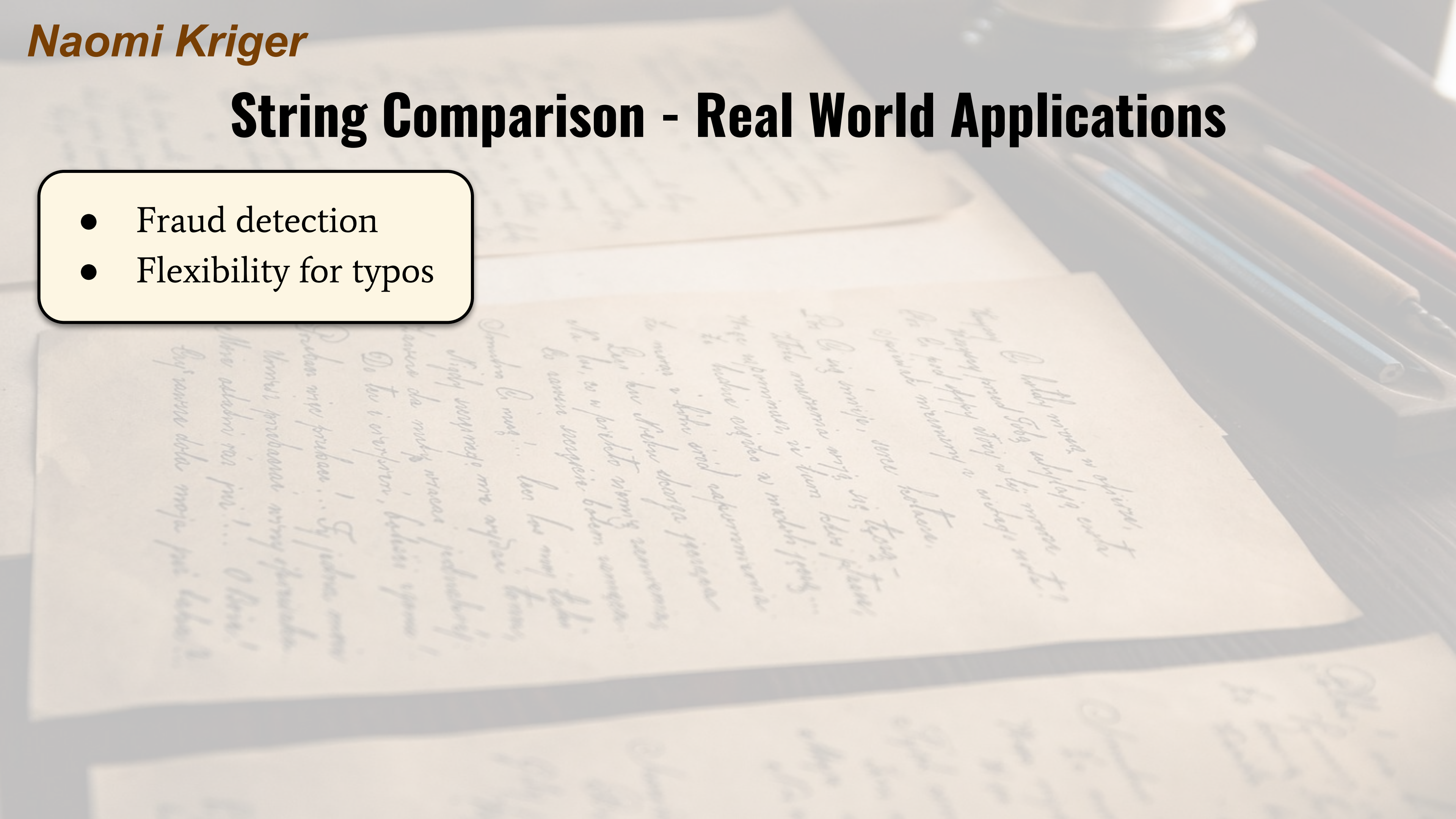
Full name: George Forge
Address: 123 Made Up Lane, Central State, US 45476

STOLEN IDENTITY

Full name: Mr. Georgie Forge
Address: 123 Made-Up ln., Central State, US 0045476

Naomi Kriger

String Comparison - Real World Applications



- Fraud detection
- Flexibility for typos

Naomi Kriger

String Comparison - Real World Applications

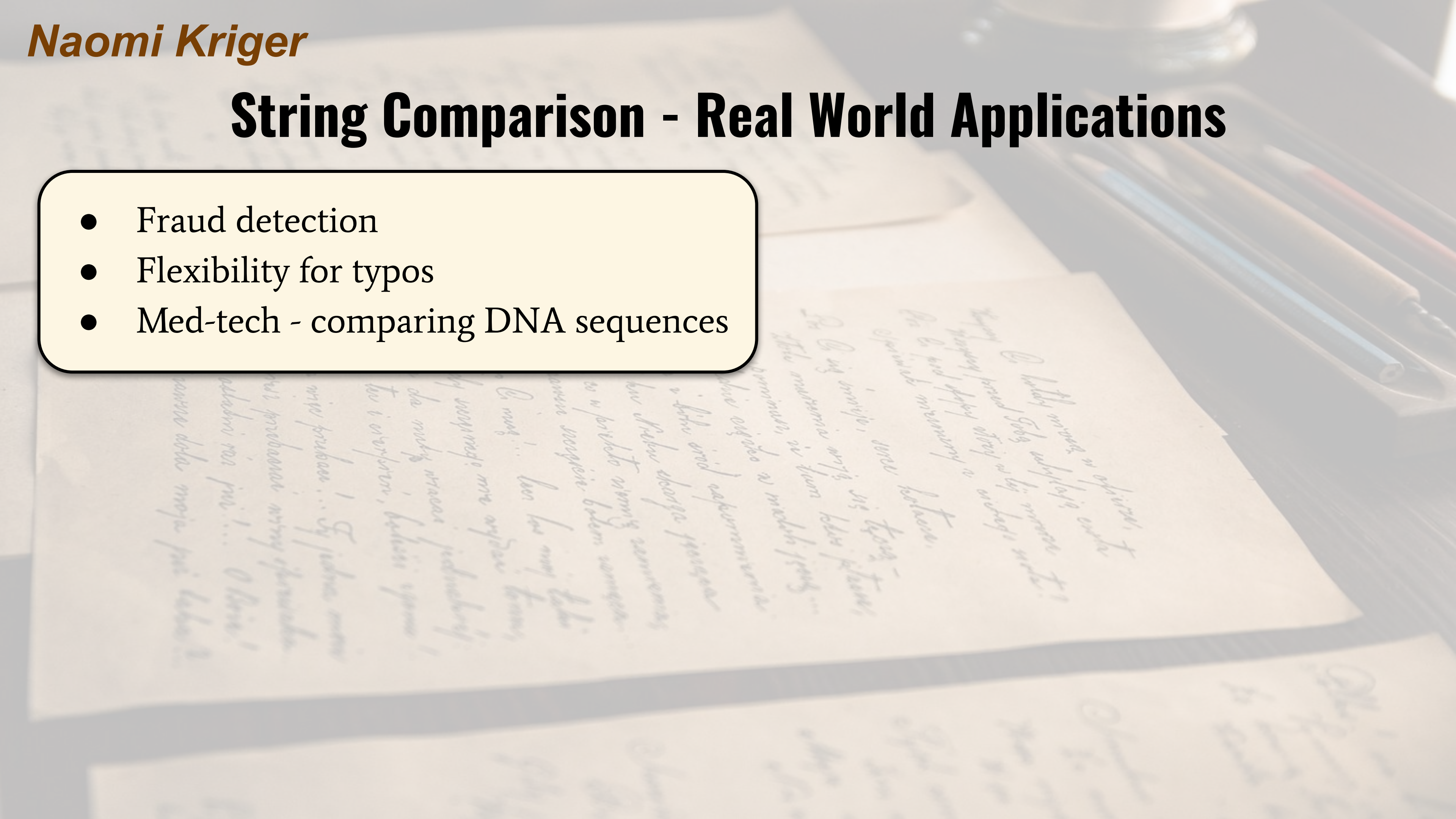
- Fraud detection
- Flexibility for typos

assessment / asessment

hazardous / hazrdus

responsibility / responsability

String Comparison - Real World Applications



- Fraud detection
- Flexibility for typos
- Med-tech - comparing DNA sequences

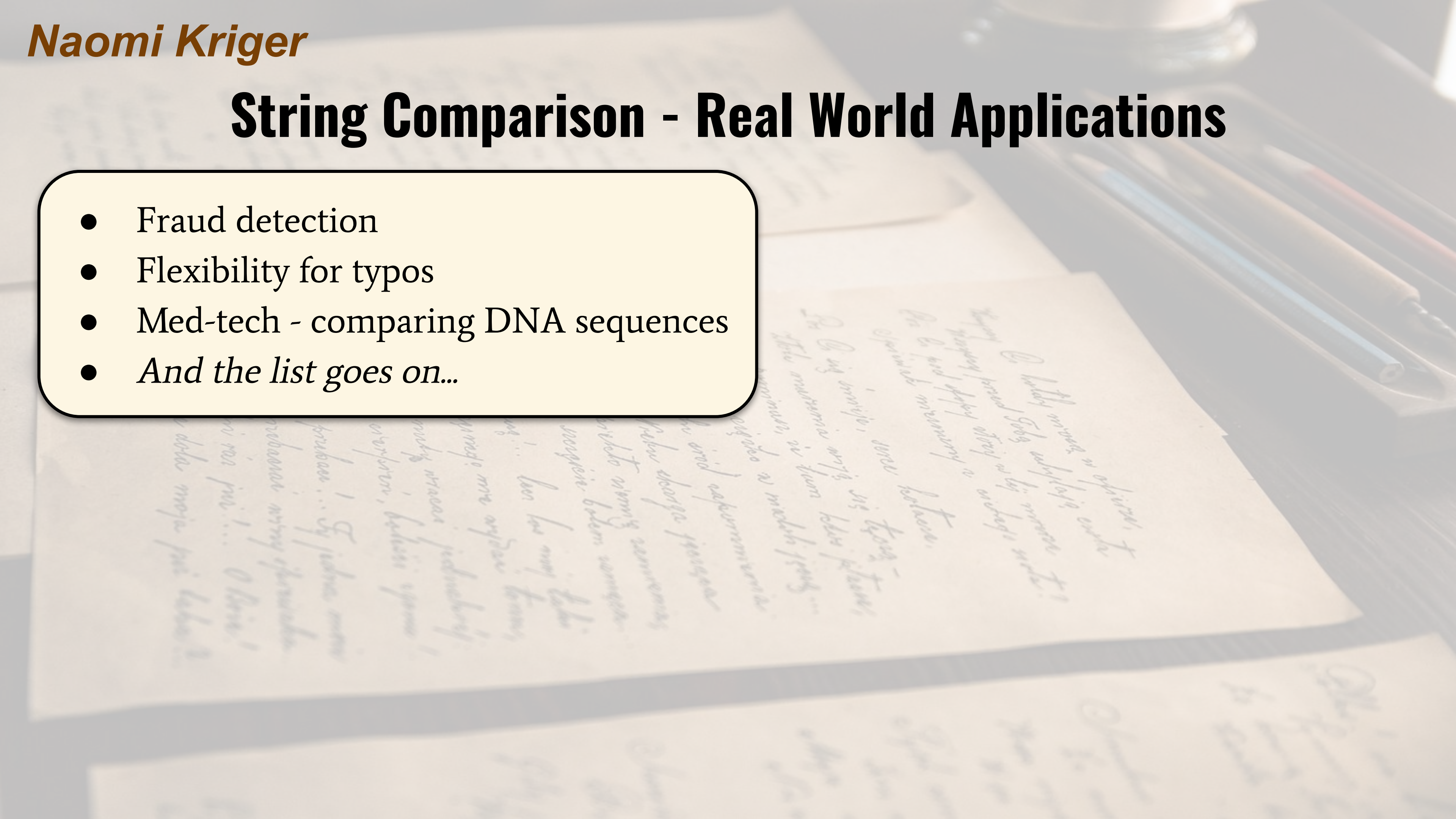
String Comparison - Real World Applications

- Fraud detection
- Flexibility for typos
- Med-tech - comparing DNA sequences

ATGACGTGGGAA
ATAACGTGGGCA

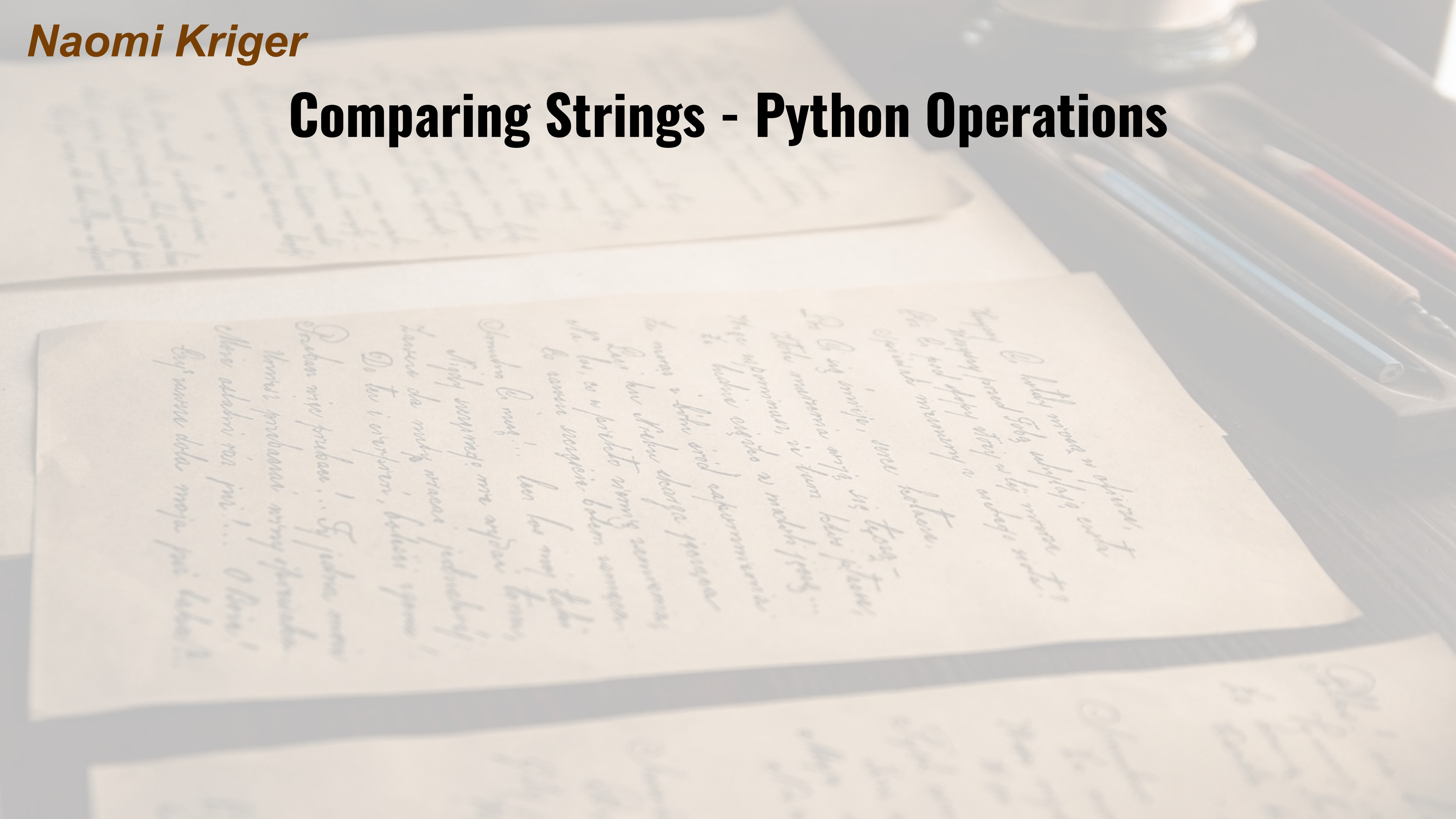
Naomi Kriger

String Comparison - Real World Applications



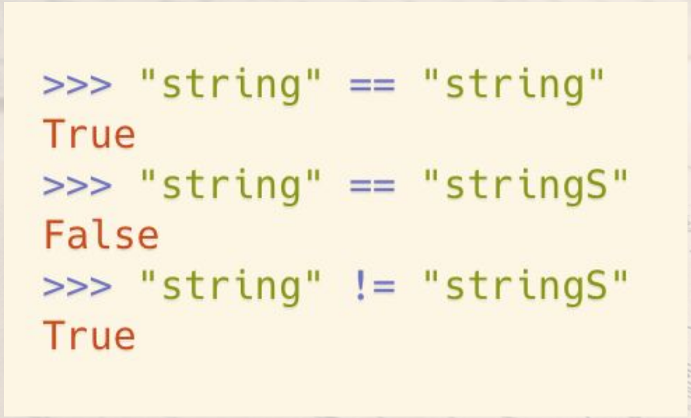
- Fraud detection
- Flexibility for typos
- Med-tech - comparing DNA sequences
- *And the list goes on...*

Comparing Strings - Python Operations



Naomi Kriger

Comparing Strings - Python Operations



```
>>> "string" == "string"
True
>>> "string" == "stringS"
False
>>> "string" != "stringS"
True
```

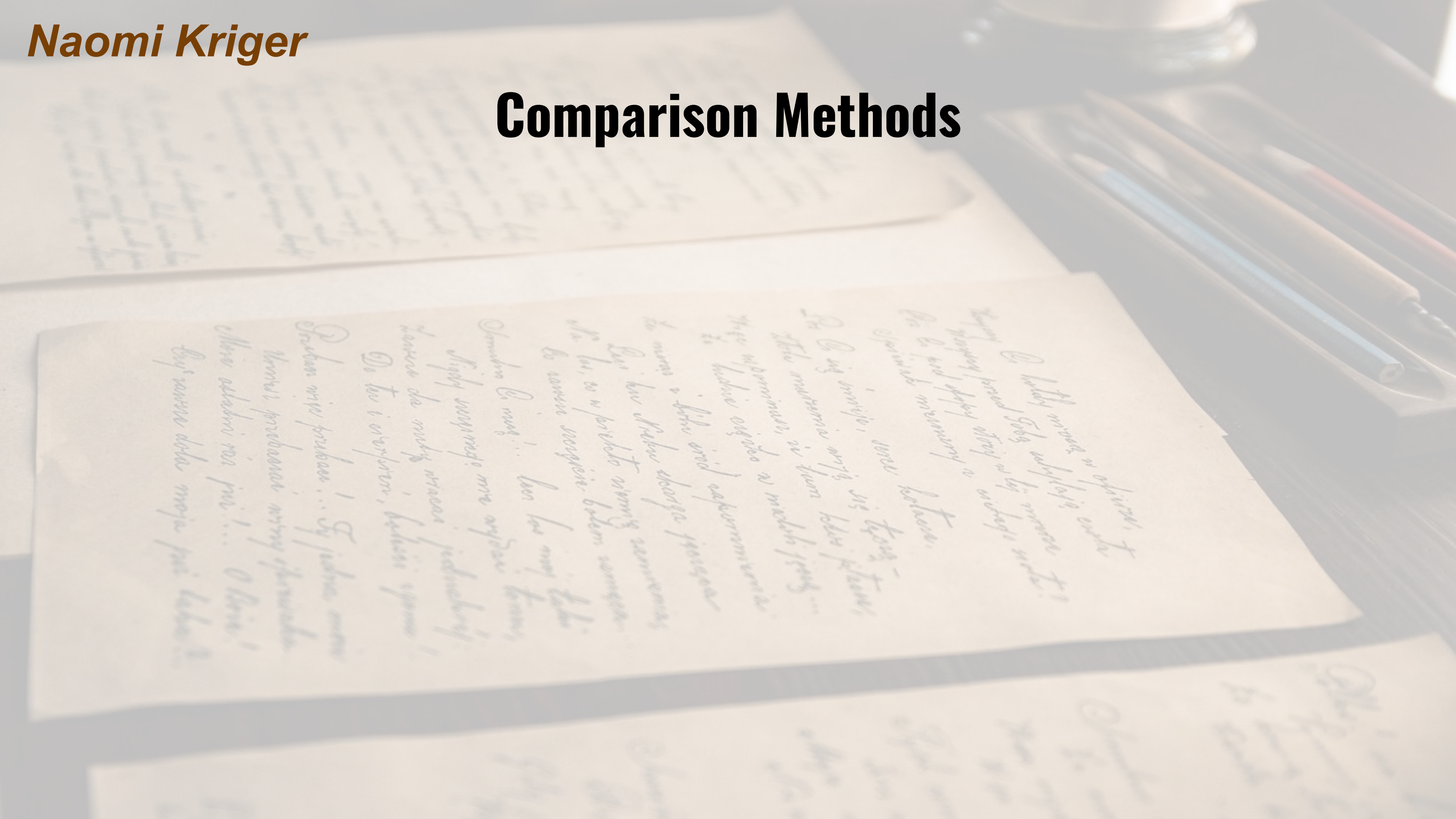
Naomi Kriger

Comparing Strings - Python Operations

```
>>> "string" == "string"
True
>>> "string" == "stringS"
False
>>> "string" != "stringS"
True
```

```
>>> a = "my new string"
>>> b = "new"
>>> a in b
False
>>> b in a
True
```

Comparison Methods



Naomi Kriger

Comparison Methods

Naomi Kriger

Comparison Methods

Jellyfish

Edit distance

Phonetic encoding

Jellyfish

Edit distance

- *Levenshtein Distance*
- *Damerau-Levenshtein Distance*

Naomi Kriger

String Metric

String Metric

Receives two strings

	s	t	r	i	n	g
s						
t						
i						
n						
g						

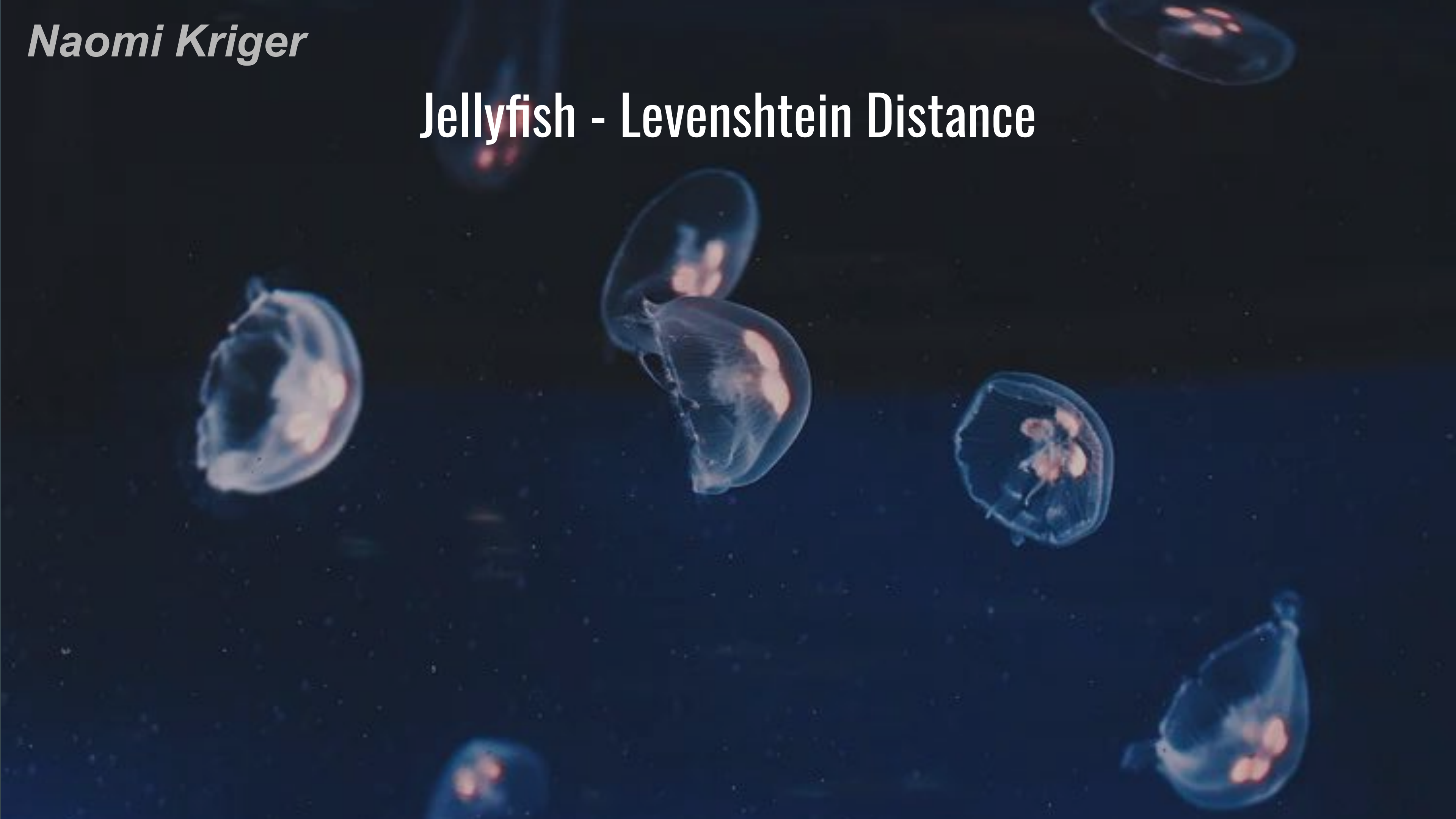
String Metric

Receives two strings
and produces a distance score

	s	t	r	i	n	g
s						
t						
i						
n						
g						

magic

distance("string", "sting") = 1



Jellyfish - Levenshtein Distance

Jellyfish - Levenshtein Distance

- Calculates the minimal steps required to convert string A to string B
- A step is one of the followings:
  - **Addition, deletion, replacement**
- Higher score → bigger difference

Jellyfish - Levenshtein Distance

```
>>> jellyfish.levenshtein_distance(“exit“, “exist“)
```

Jellyfish - Levenshtein Distance

```
>>> jellyfish.levenshtein_distance(“exit“, “exist“)
1
```

Jellyfish - Levenshtein Distance

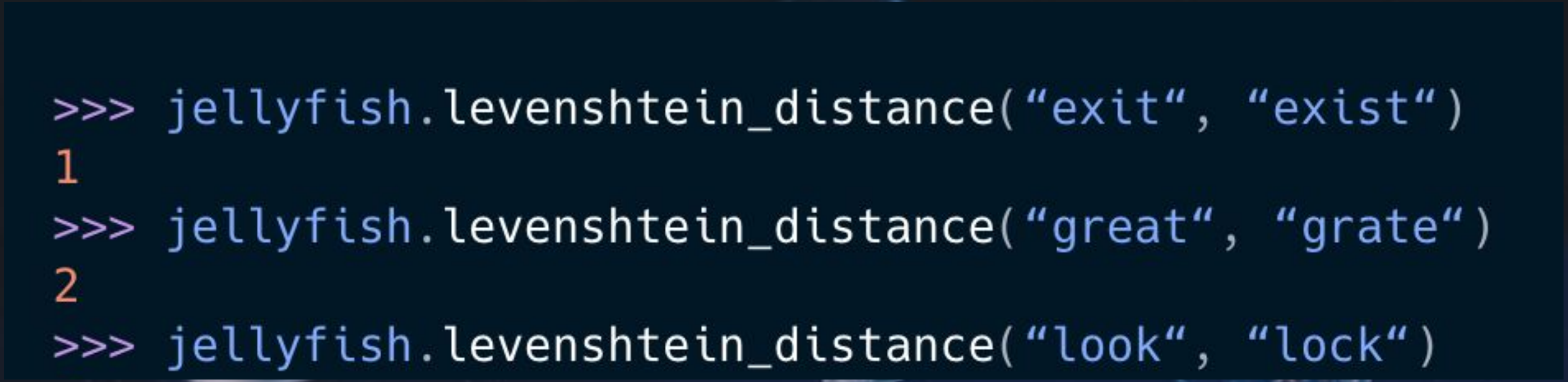
```
>>> jellyfish.levenshtein_distance(“exit“, “exist“)
1
>>> jellyfish.levenshtein_distance(“great“, “grate“)
```

Naomi Kriger

Jellyfish - Levenshtein Distance

```
>>> jellyfish.levenshtein_distance(“exit“, “exist“)
1
>>> jellyfish.levenshtein_distance(“great“, “grate“)
2
```

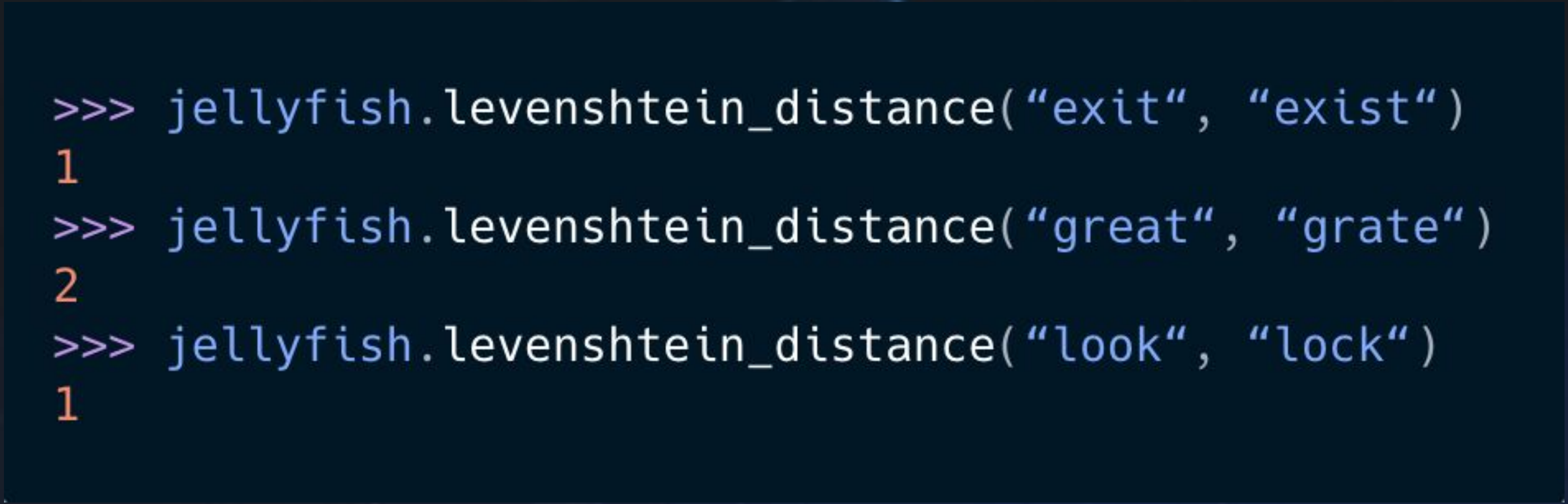
Jellyfish - Levenshtein Distance



```
>>> jellyfish.levenshtein_distance(“exit“, “exist“)
1
>>> jellyfish.levenshtein_distance(“great“, “grate“)
2
>>> jellyfish.levenshtein_distance(“look“, “lock“)
```

Naomi Kriger

Jellyfish - Levenshtein Distance



```
>>> jellyfish.levenshtein_distance(“exit“, “exist“)
1
>>> jellyfish.levenshtein_distance(“great“, “grate“)
2
>>> jellyfish.levenshtein_distance(“look“, “lock“)
1
```

Jellyfish - Damerau-Levenshtein Distance

- Calculates the minimal steps required to convert string A to string B
- Higher score → bigger difference
- A step is one of the followings:
  - **Addition, deletion, replacement**

Naomi Kriger

Jellyfish - Damerau-Levenshtein Distance

- Calculates the minimal steps required to convert string A to string B
- Higher score → bigger difference
- A step is one of the followings:
  - **Addition, deletion, replacement**
- Counts a swap of two adjacent characters as a **single** step, unlike Levenshtein Distance which counts them as two steps

Jellyfish - Damerau-Levenshtein Distance

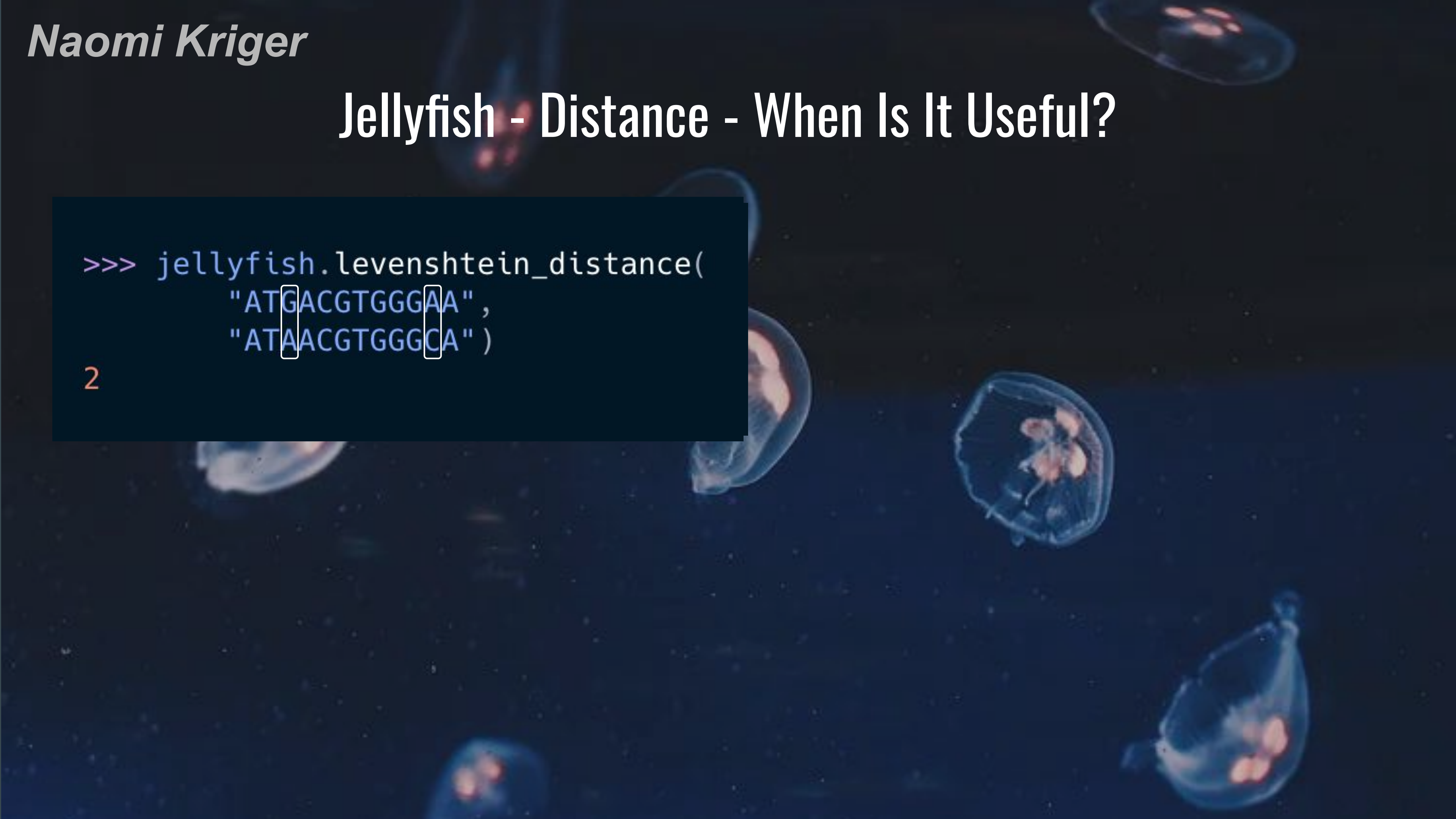
```
>>> jellyfish.damerau_levenshtein_distance("swap", "sawp")
1
>>> jellyfish.levenshtein_distance("swap", "sawp")
2
```

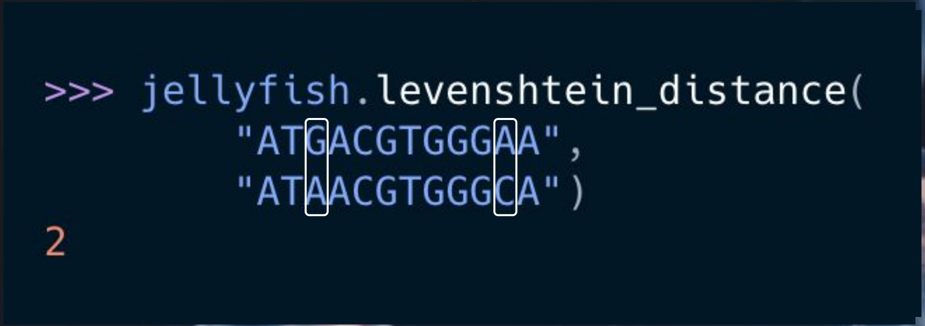
Naomi Kriger

Jellyfish - Distance - When Is It Useful?

Naomi Kriger

Jellyfish - Distance - When Is It Useful?





```
>>> jellyfish.levenshtein_distance(
        "ATGACGTGGGAA",
        "ATAACGTGGGCA")
2
```

Naomi Kriger

Jellyfish - Distance - When Is It Useful?

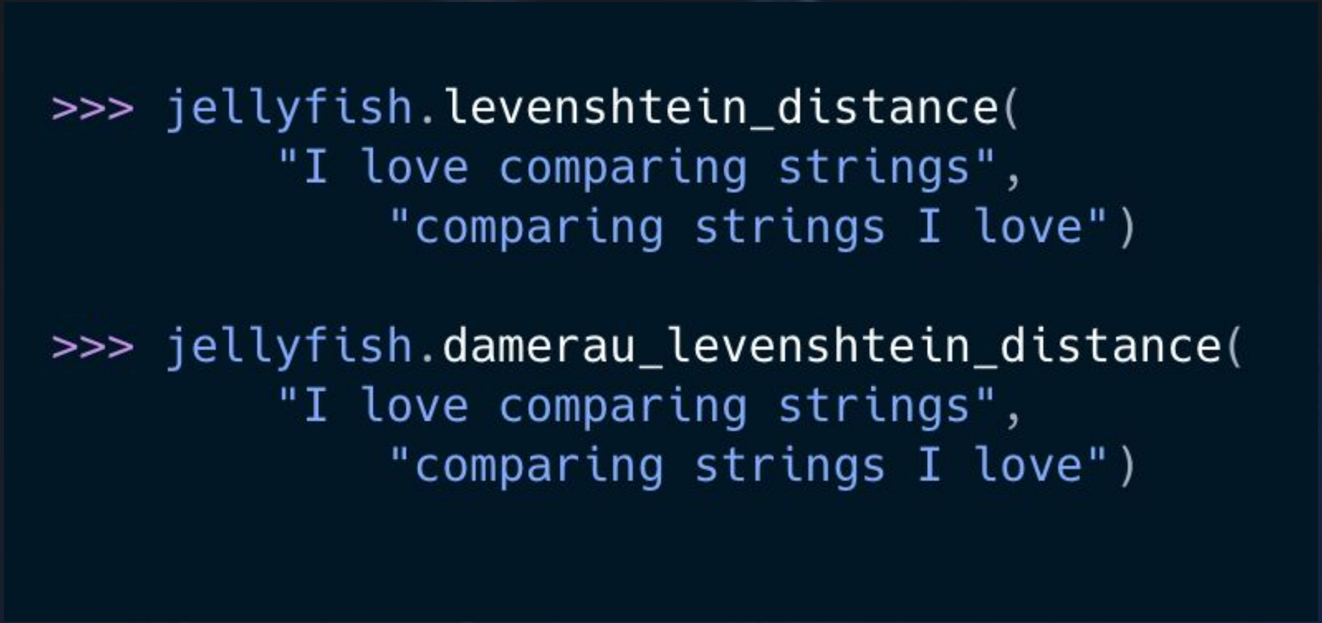
```
>>> jellyfish.levenshtein_distance("Mr. Bean", "Mr Bean")
1
>>> jellyfish.damerau_levenshtein_distance("Johnny Depp", "Jhonny Depp")
1
```

Naomi Kriger

Jellyfish - Distance - When Is It NOT Useful?

Naomi Kriger

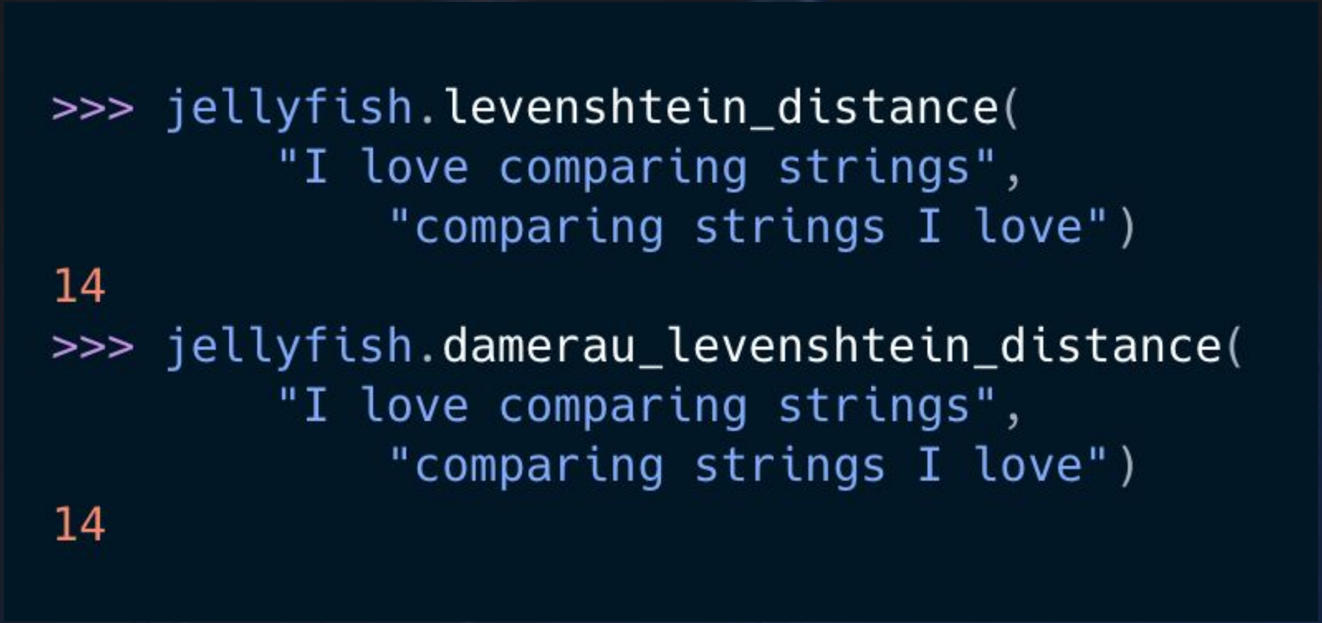
Jellyfish - Distance - When Is It NOT Useful?



```
>>> jellyfish.levenshtein_distance(
        "I love comparing strings",
            "comparing strings I love")

>>> jellyfish.damerau_levenshtein_distance(
        "I love comparing strings",
            "comparing strings I love")
```

Jellyfish - Distance - When Is It NOT Useful?



```
>>> jellyfish.levenshtein_distance(
        "I love comparing strings",
            "comparing strings I love")
14
>>> jellyfish.damerau_levenshtein_distance(
        "I love comparing strings",
            "comparing strings I love")
14
```

Naomi Kriger

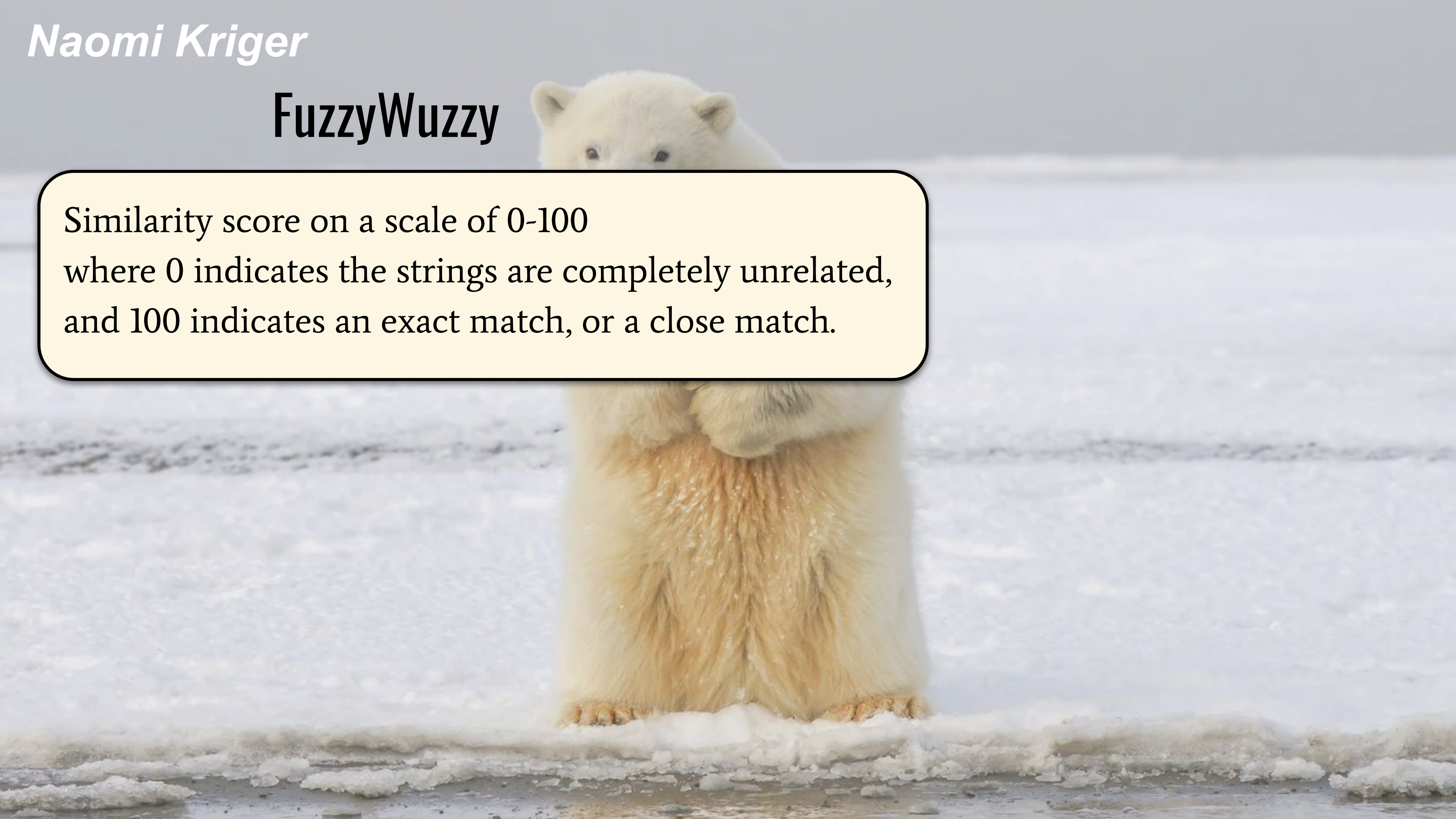
Fuzzy Wuzzy was a bear.
Fuzzy Wuzzy had no hair.
So Fuzzy Wuzzy
wasn't very fuzzy,
was he?

Naomi Kriger

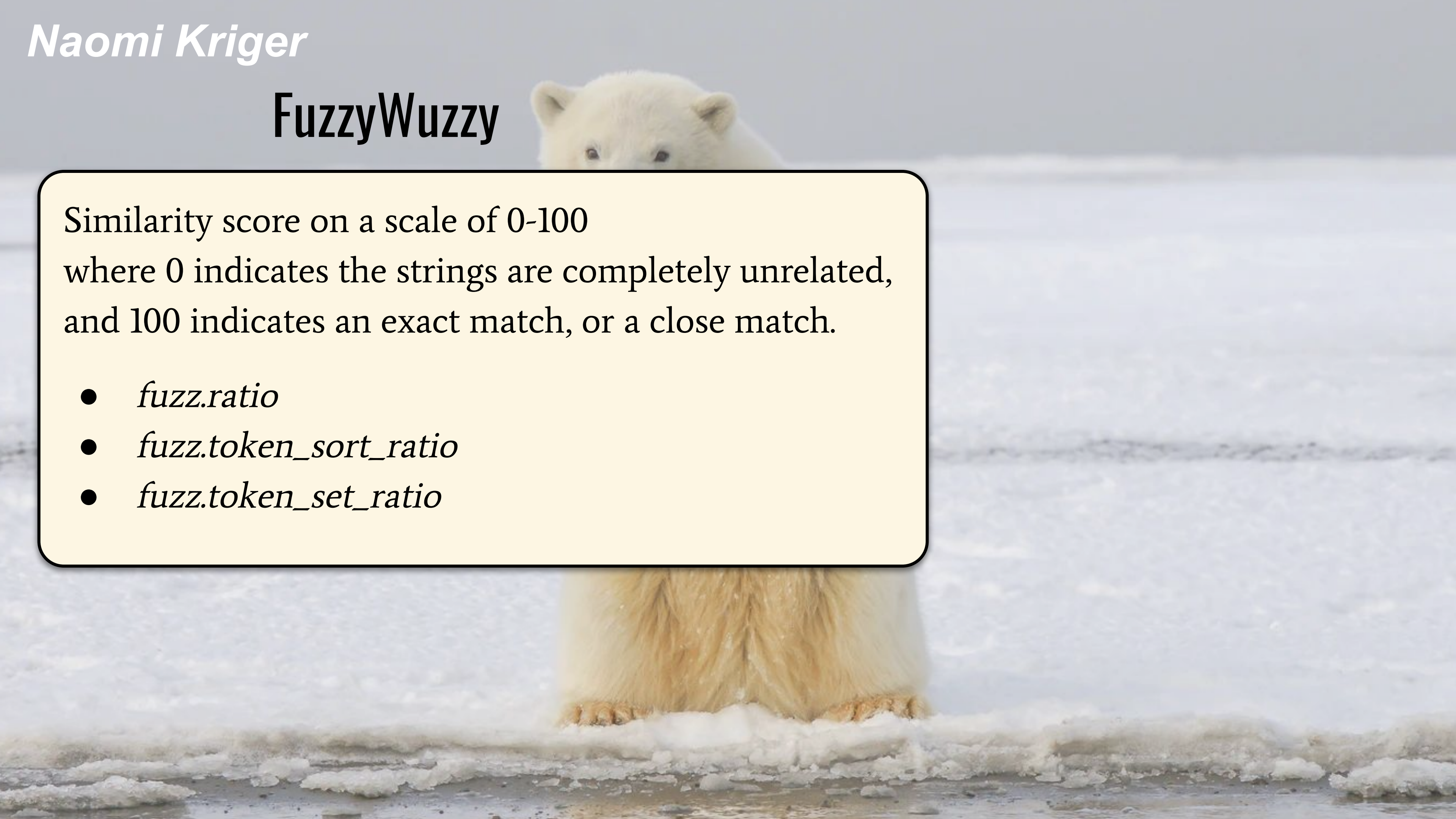
FuzzyWuzzy

Naomi Kriger

FuzzyWuzzy



Similarity score on a scale of 0-100
where 0 indicates the strings are completely unrelated,
and 100 indicates an exact match, or a close match.



FuzzyWuzzy

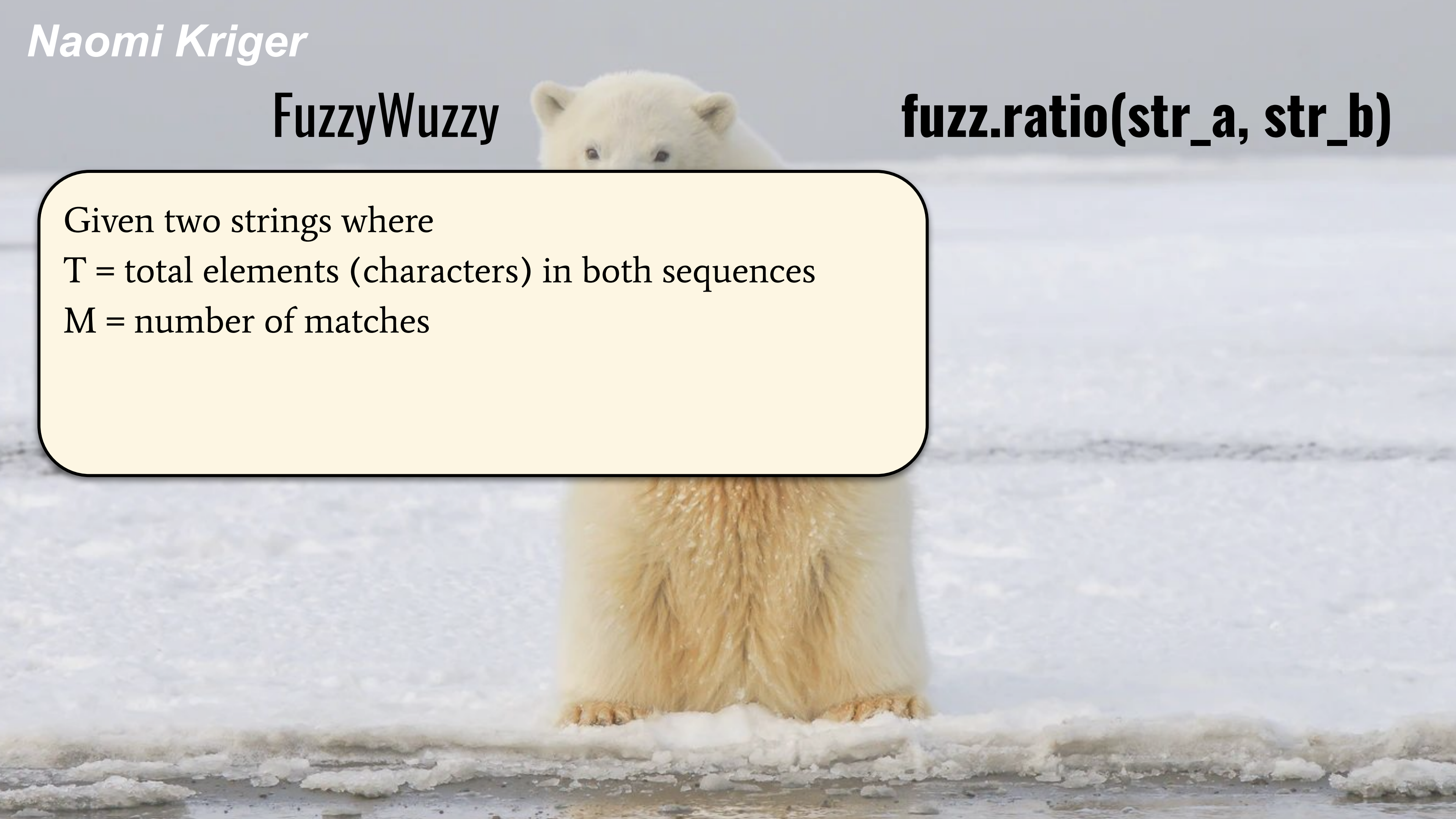
Similarity score on a scale of 0-100
where 0 indicates the strings are completely unrelated,
and 100 indicates an exact match, or a close match.

- *fuzz.ratio*
- *fuzz.token_sort_ratio*
- *fuzz.token_set_ratio*

Naomi Kriger

FuzzyWuzzy

`fuzz.ratio(str_a, str_b)`



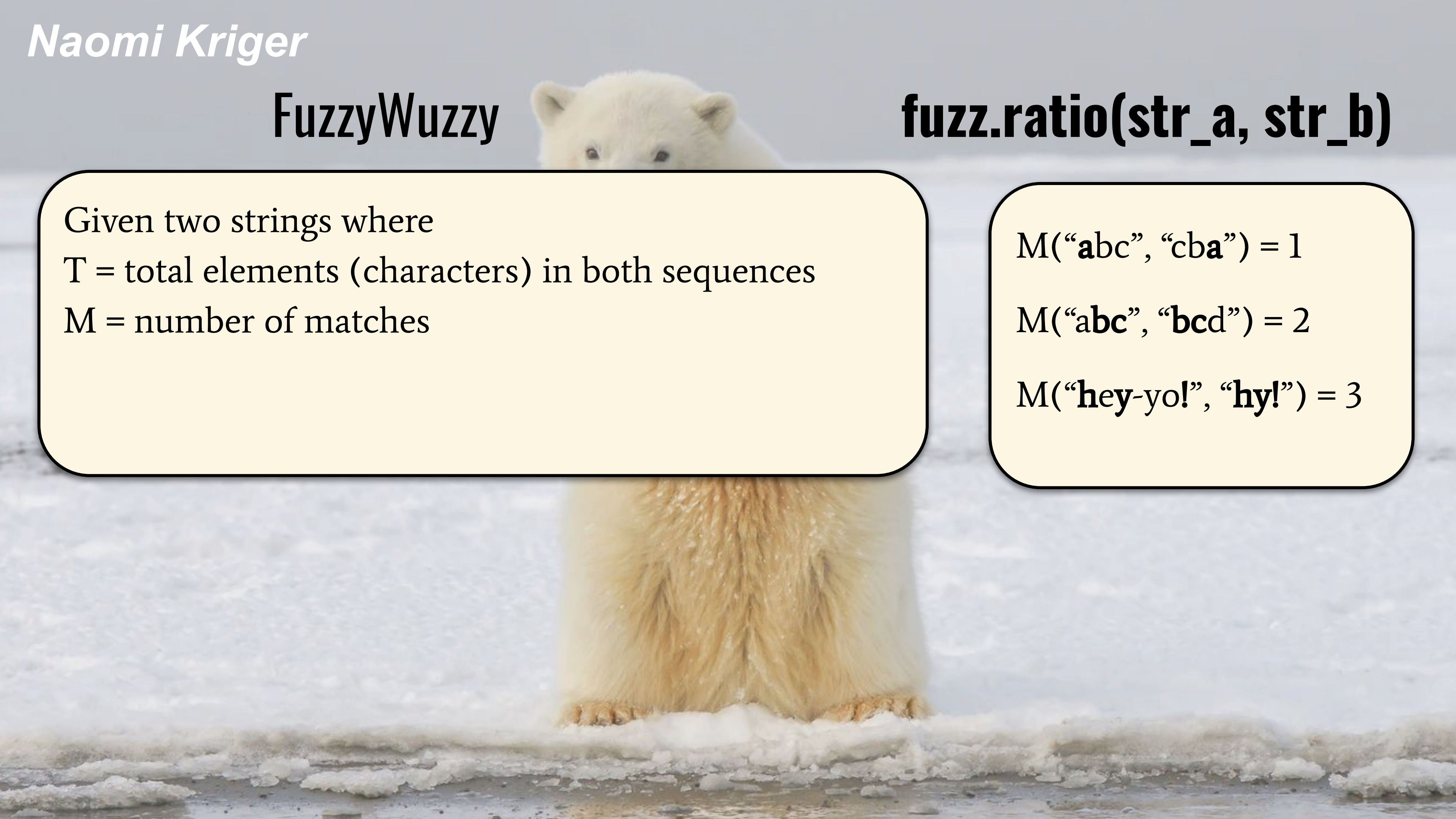
FuzzyWuzzy

fuzz.ratio(str_a, str_b)

Given two strings where

T = total elements (characters) in both sequences

M = number of matches



FuzzyWuzzy

fuzz.ratio(str_a, str_b)

Given two strings where

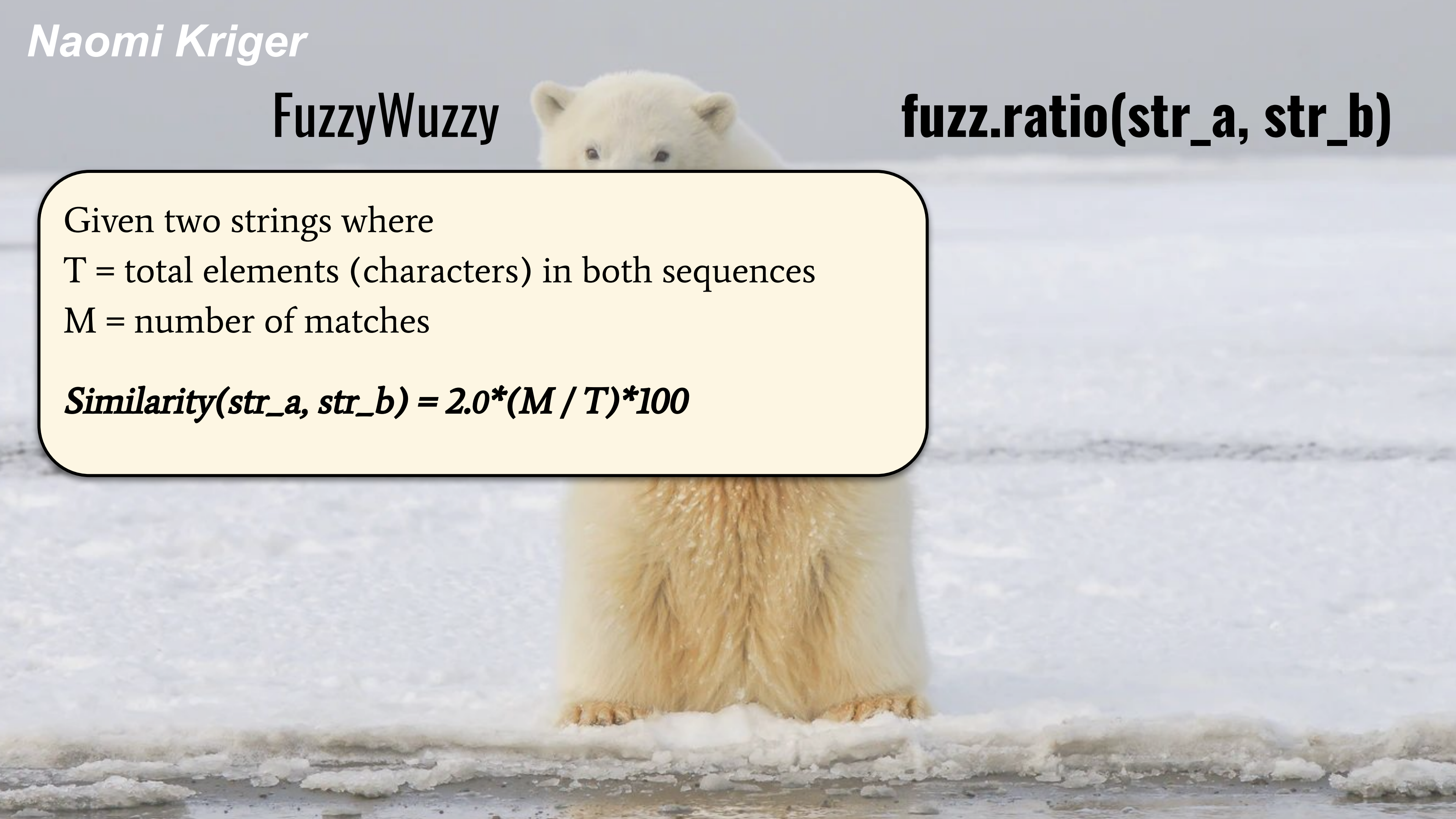
T = total elements (characters) in both sequences

M = number of matches

M(“**a**bc”, “cb**a**”) = 1

M(“a**bc**”, “**bc**d”) = 2

M(“**hey**-yo**!**”, “**hy!**”) = 3



FuzzyWuzzy

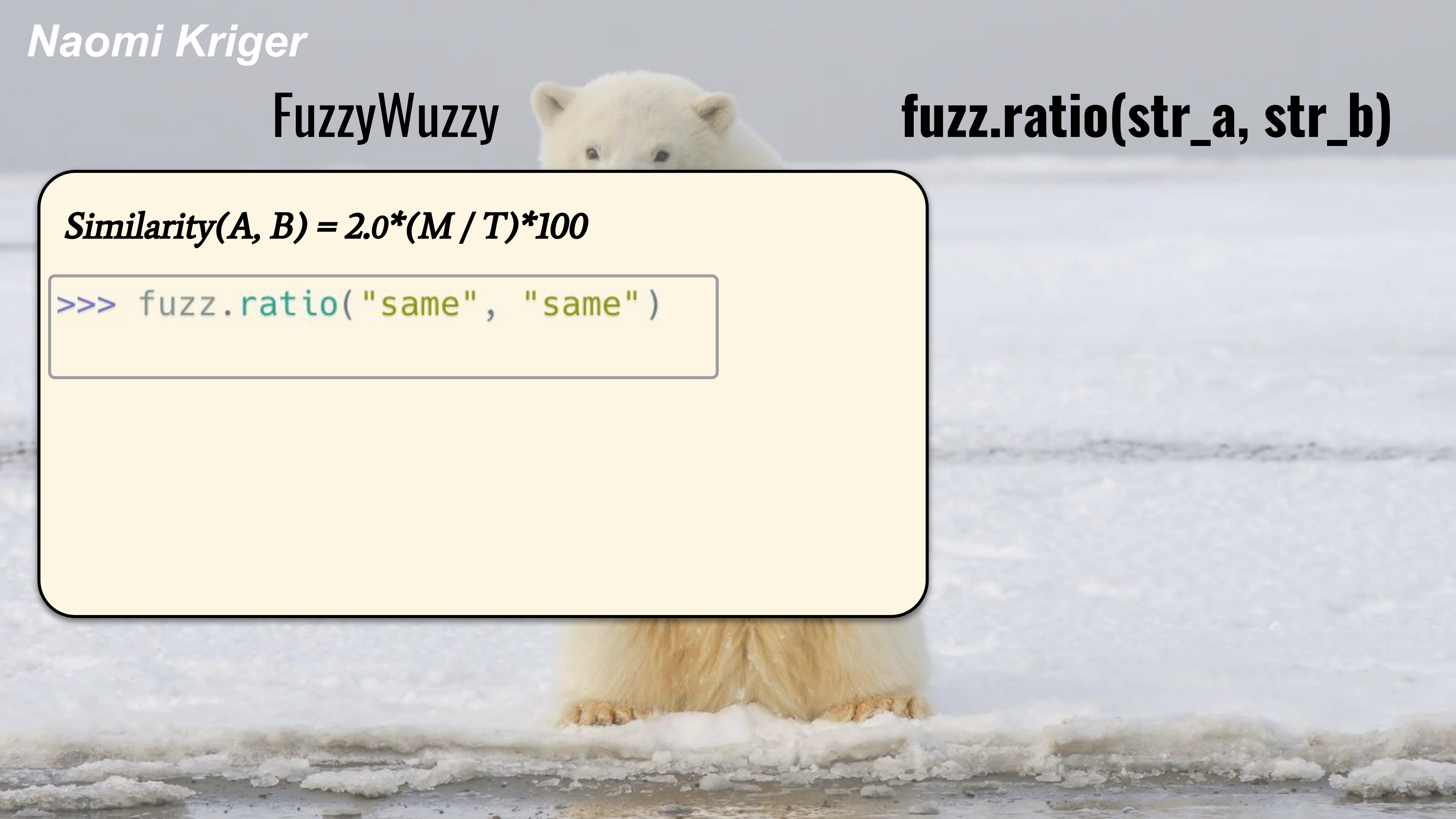
fuzz.ratio(str_a, str_b)

Given two strings where

T = total elements (characters) in both sequences

M = number of matches

Similarity(str_a, str_b) = 2.0*(M / T)*100



Naomi Kriger

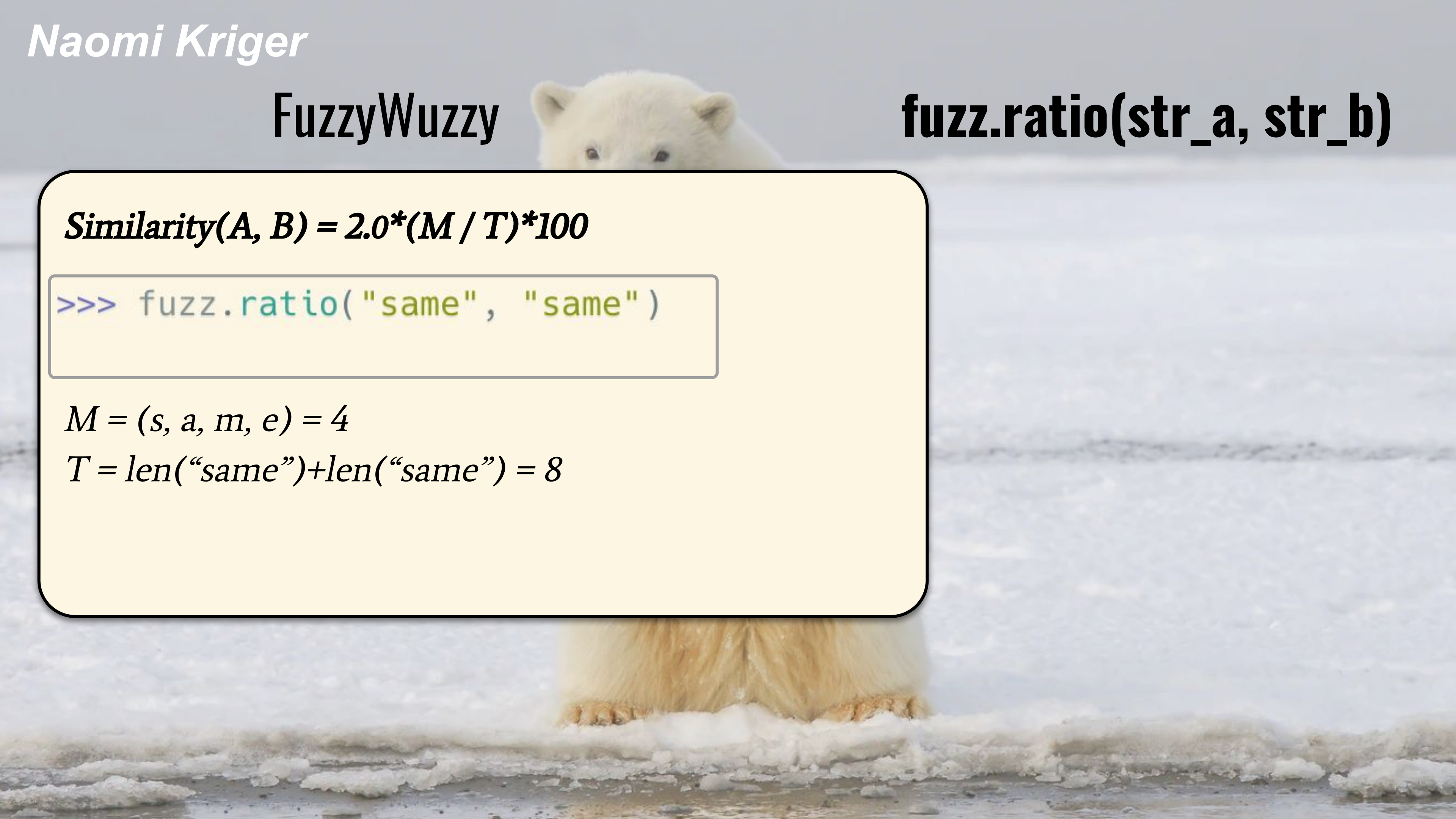
FuzzyWuzzy

fuzz.ratio(str_a, str_b)

Similarity(A, B) = 2.0(M / T)*100*

```
>>> fuzz.ratio("same", "same")
```

Naomi Kriger



FuzzyWuzzy

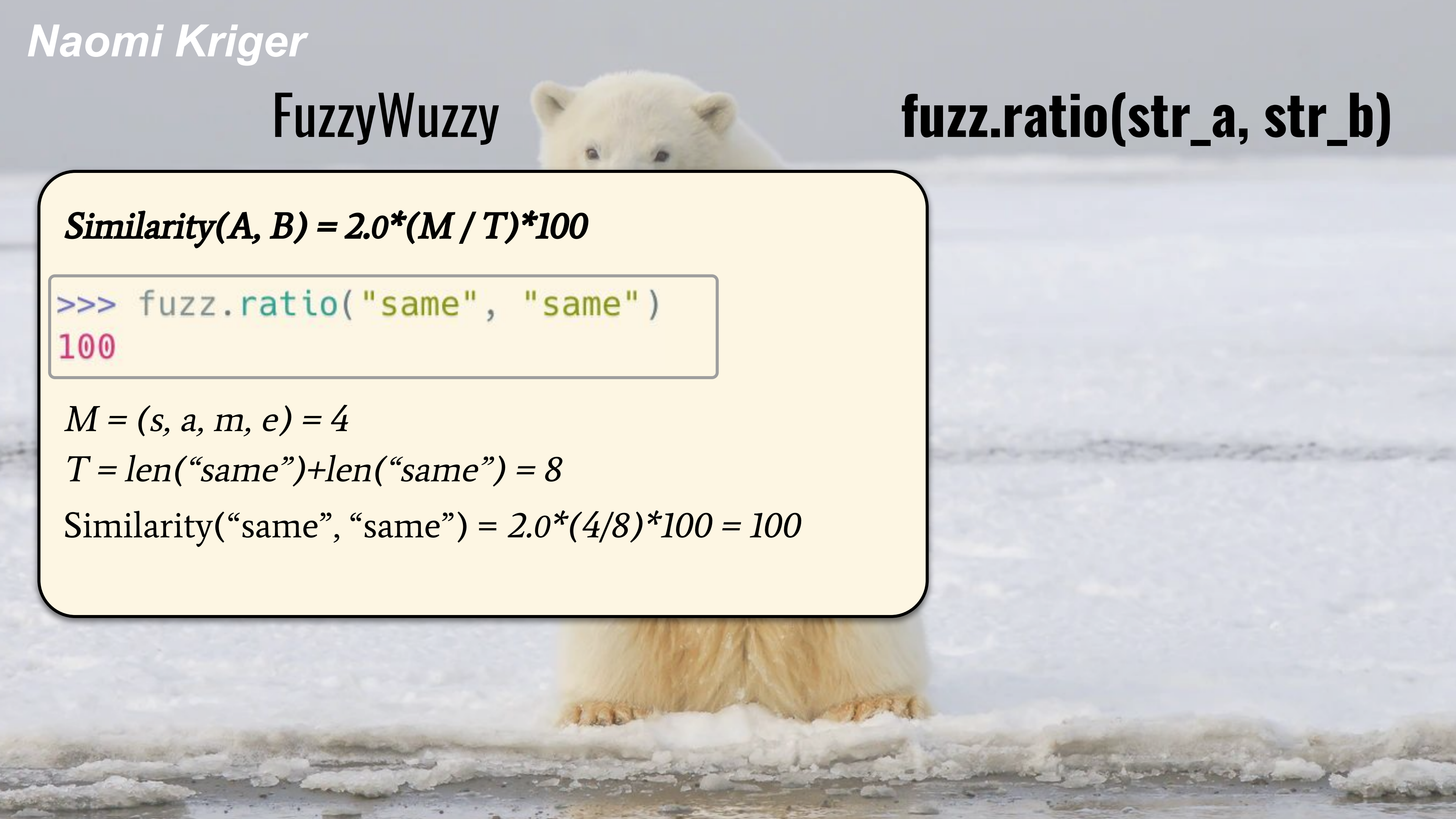
fuzz.ratio(str_a, str_b)

Similarity(A, B) = 2.0*(M / T)*100

```
>>> fuzz.ratio("same", "same")
```

M = (s, a, m, e) = 4

T = len("same")+len("same") = 8



Naomi Kriger

FuzzyWuzzy

fuzz.ratio(str_a, str_b)

Similarity(A, B) = 2.0*(M / T)*100

```
>>> fuzz.ratio("same", "same")
100
```

M = (s, a, m, e) = 4

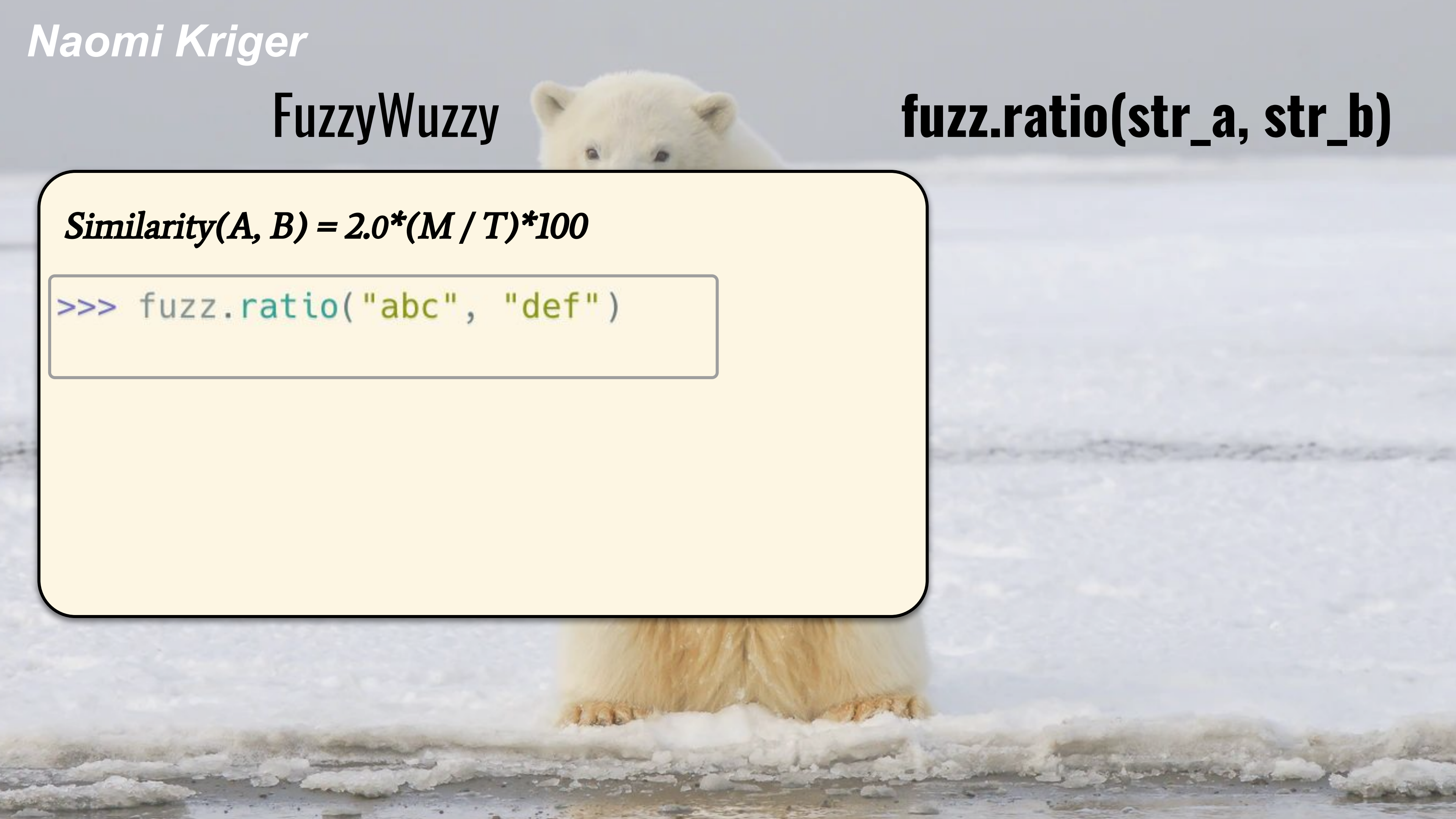
T = len("same")+len("same") = 8

Similarity("same", "same") = *2.0*(4/8)*100 = 100*

Naomi Kriger

FuzzyWuzzy

fuzz.ratio(str_a, str_b)

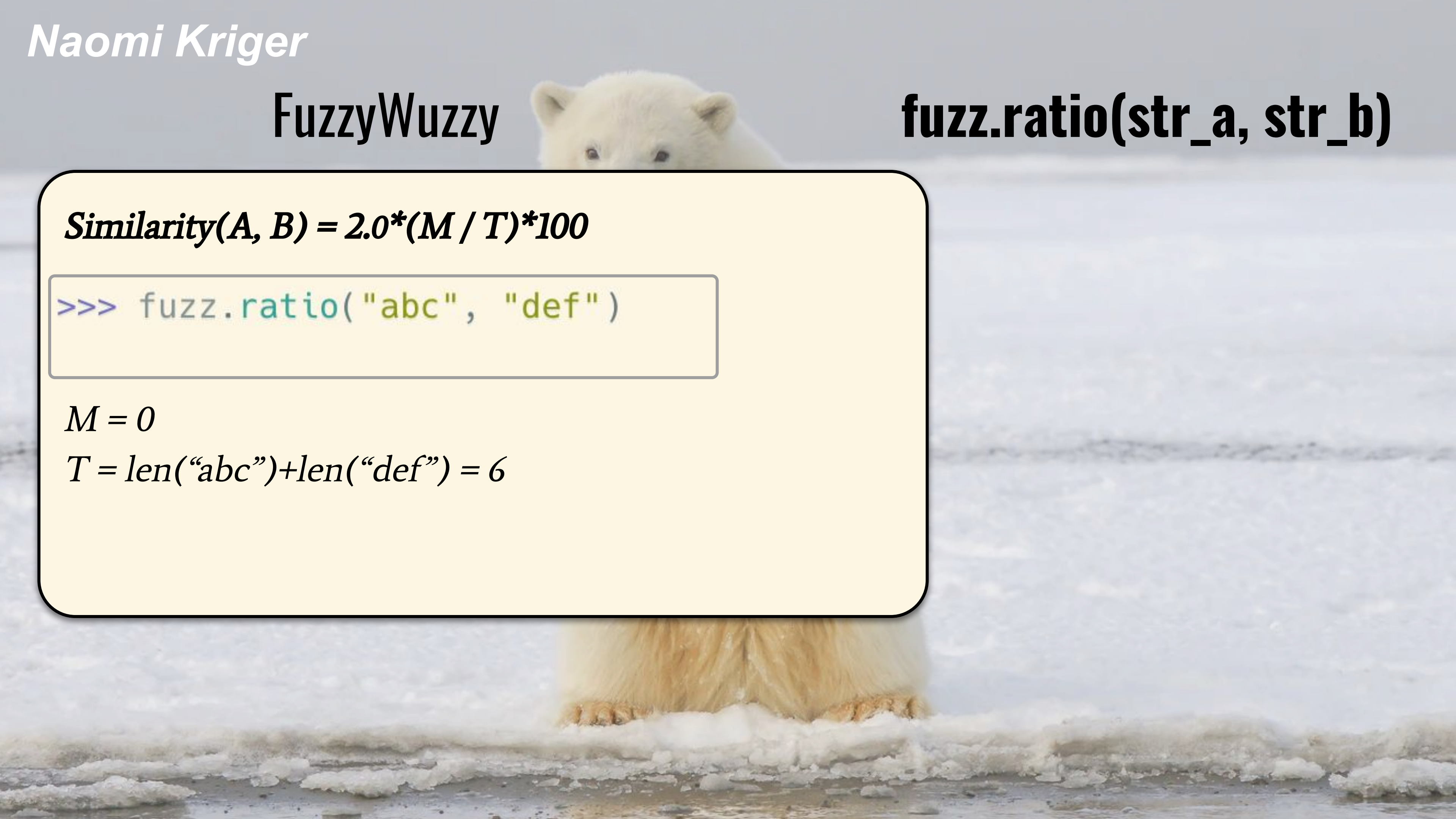


Similarity(A, B) = 2.0(M / T)*100*

```
>>> fuzz.ratio("abc", "def")
```

FuzzyWuzzy

fuzz.ratio(str_a, str_b)

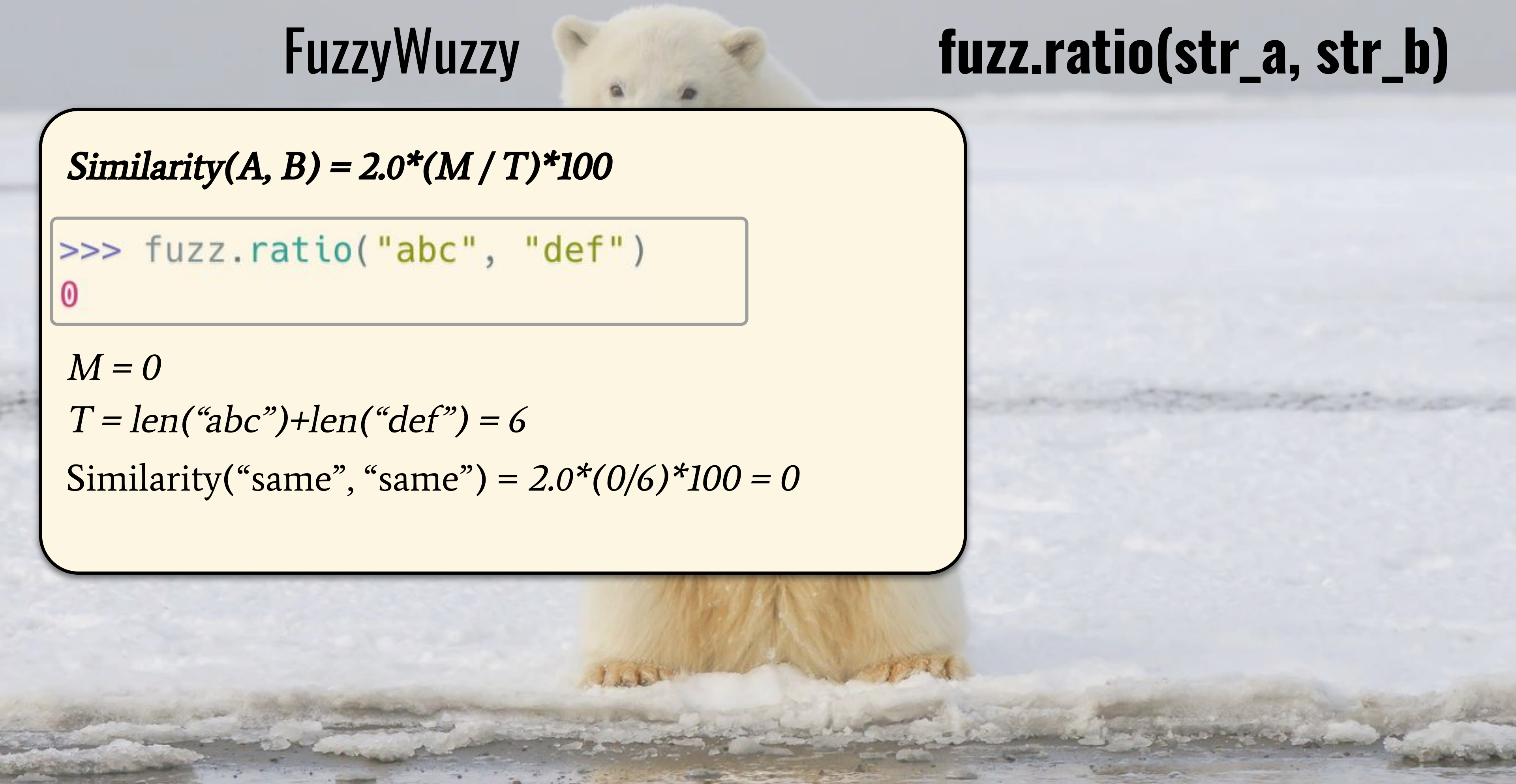


Similarity(A, B) = 2.0*(M / T)*100

```
>>> fuzz.ratio("abc", "def")
```

$M = 0$

$T = len(“abc”)+len(“def”) = 6$



FuzzyWuzzy

fuzz.ratio(str_a, str_b)

Similarity(A, B) = 2.0*(M / T)*100

```
>>> fuzz.ratio("abc", "def")
0
```

M = 0

T = len("abc")+len("def") = 6

Similarity("same", "same") = *2.0*(0/6)*100 = 0*

FuzzyWuzzy

fuzz.ratio(str_a, str_b)

Similarity(A, B) = 2.0(M / T)*100*

```
>>> fuzz.ratio("great", "green")
```

Naomi Kriger

FuzzyWuzzy

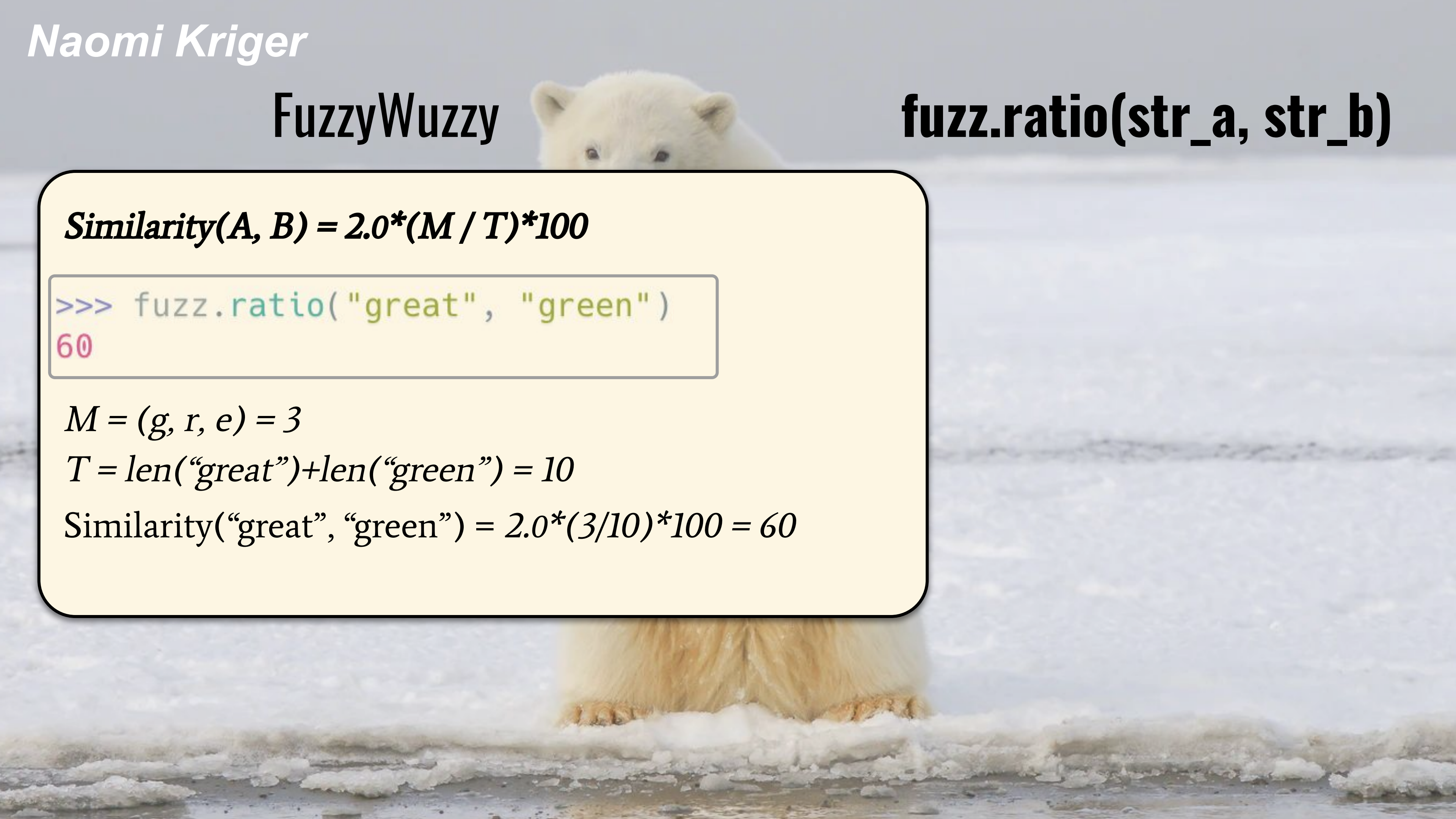
fuzz.ratio(str_a, str_b)

Similarity(A, B) = 2.0*(M / T)*100

```
>>> fuzz.ratio("great", "green")
```

M = (g, r, e) = 3

T = len("great")+len("green") = 10



Naomi Kriger

FuzzyWuzzy

fuzz.ratio(str_a, str_b)

Similarity(A, B) = 2.0*(M / T)*100

```
>>> fuzz.ratio("great", "green")
60
```

M = (g, r, e) = 3

T = len("great")+len("green") = 10

Similarity("great", "green") = *2.0*(3/10)*100 = 60*

FuzzyWuzzy

fuzz.ratio(str_a, str_b)

```
>>> fuzz.ratio("1943 Evergreen Lane Gardena California 90247",
               "#1943 Evergreen Lane Gardena, California 0090247")

>>> fuzz.ratio("a", "abcde")
```

Naomi Kriger

FuzzyWuzzy

fuzz.ratio(str_a, str_b)

```
>>> fuzz.ratio("1943 Evergreen Lane Gardena California 90247",
               "#1943 Evergreen Lane Gardena, California 0090247")

>>> fuzz.ratio("a", "abcde")
```

Naomi Kriger

FuzzyWuzzy

fuzz.ratio(str_a, str_b)

```
>>> fuzz.ratio("1943 Evergreen Lane Gardena California 90247",
                    "#1943 Evergreen Lane Gardena, California 0090247")

>>> fuzz.ratio("a", "abcde")
```

```
>>> jellyfish.levenshtein_distance(
"1943 Evergreen Lane Gardena California 90247",
                "#1943 Evergreen Lane Gardena, California 0090247")
4

>>> jellyfish.levenshtein_distance("a", "abcde")
4
```

Naomi Kriger

FuzzyWuzzy

fuzz.ratio(str_a, str_b)

```
>>> fuzz.ratio("1943 Evergreen Lane Gardena California 90247",
               "#1943 Evergreen Lane Gardena, California 0090247")
96
>>> fuzz.ratio("a", "abcde")
33
```

FuzzyWuzzy

fuzz.ratio(str_a, str_b)

```
>>> fuzz.ratio("1943 Evergreen Lane Gardena California 90247",
                  "#1943 Evergreen Lane Gardena, California 0090247")
96
>>> fuzz.ratio("a", "abcde")
33
>>> fuzz.ratio("A", "a")
0
```

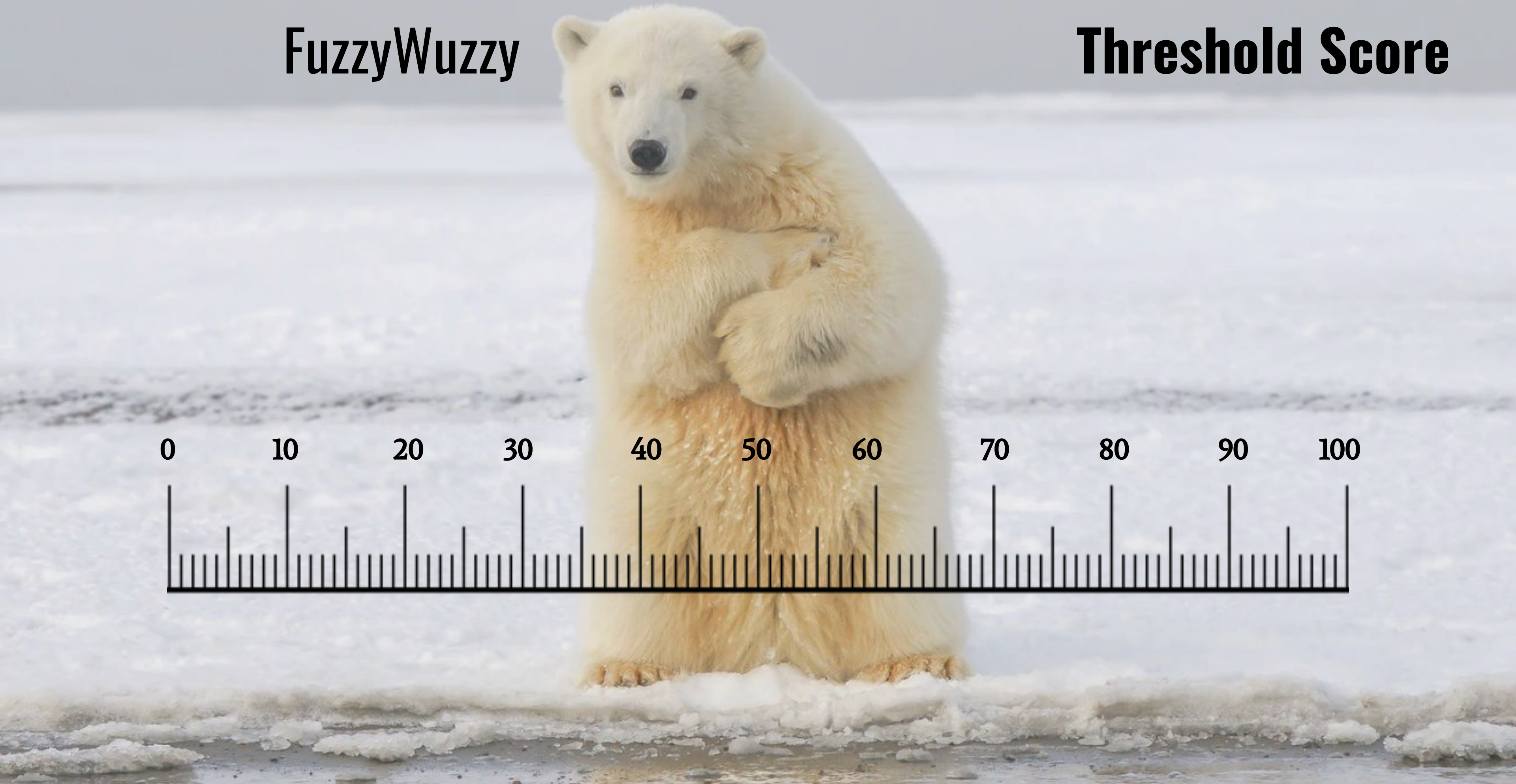
Naomi Kriger

FuzzyWuzzy

Threshold Score

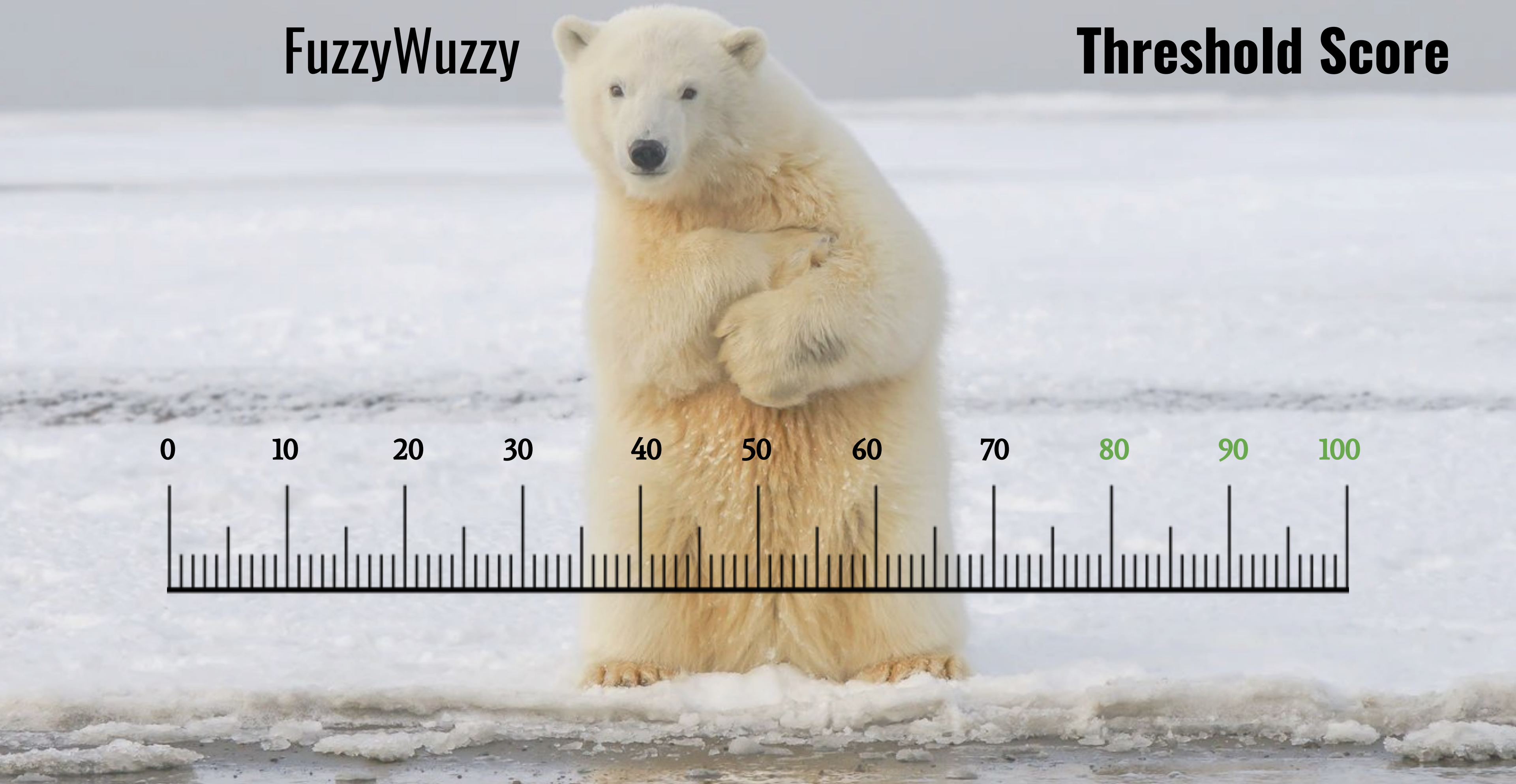
Naomi Kriger

FuzzyWuzzy Threshold Score



Naomi Kriger

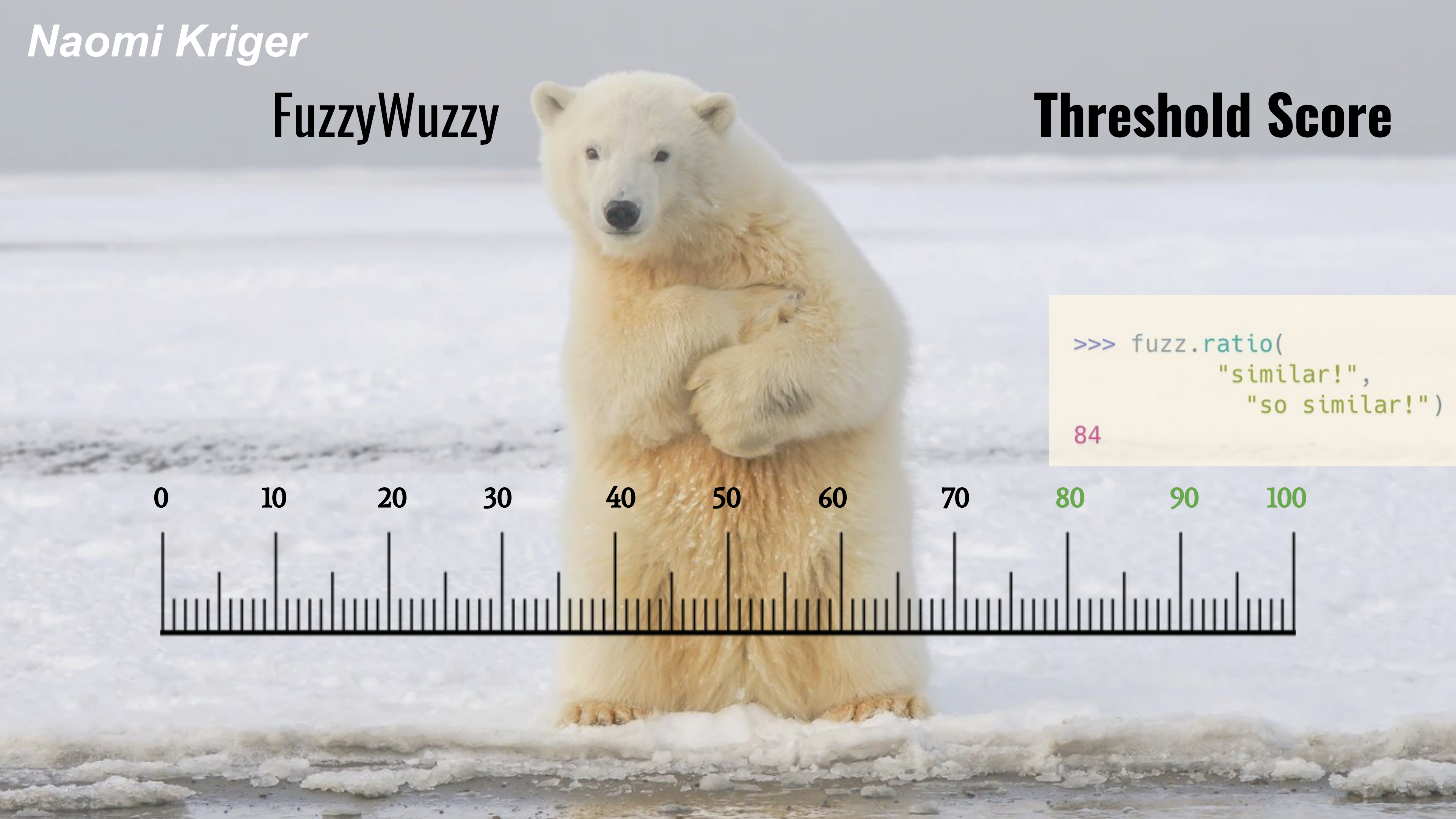
FuzzyWuzzy Threshold Score



Naomi Kriger

FuzzyWuzzy

Threshold Score

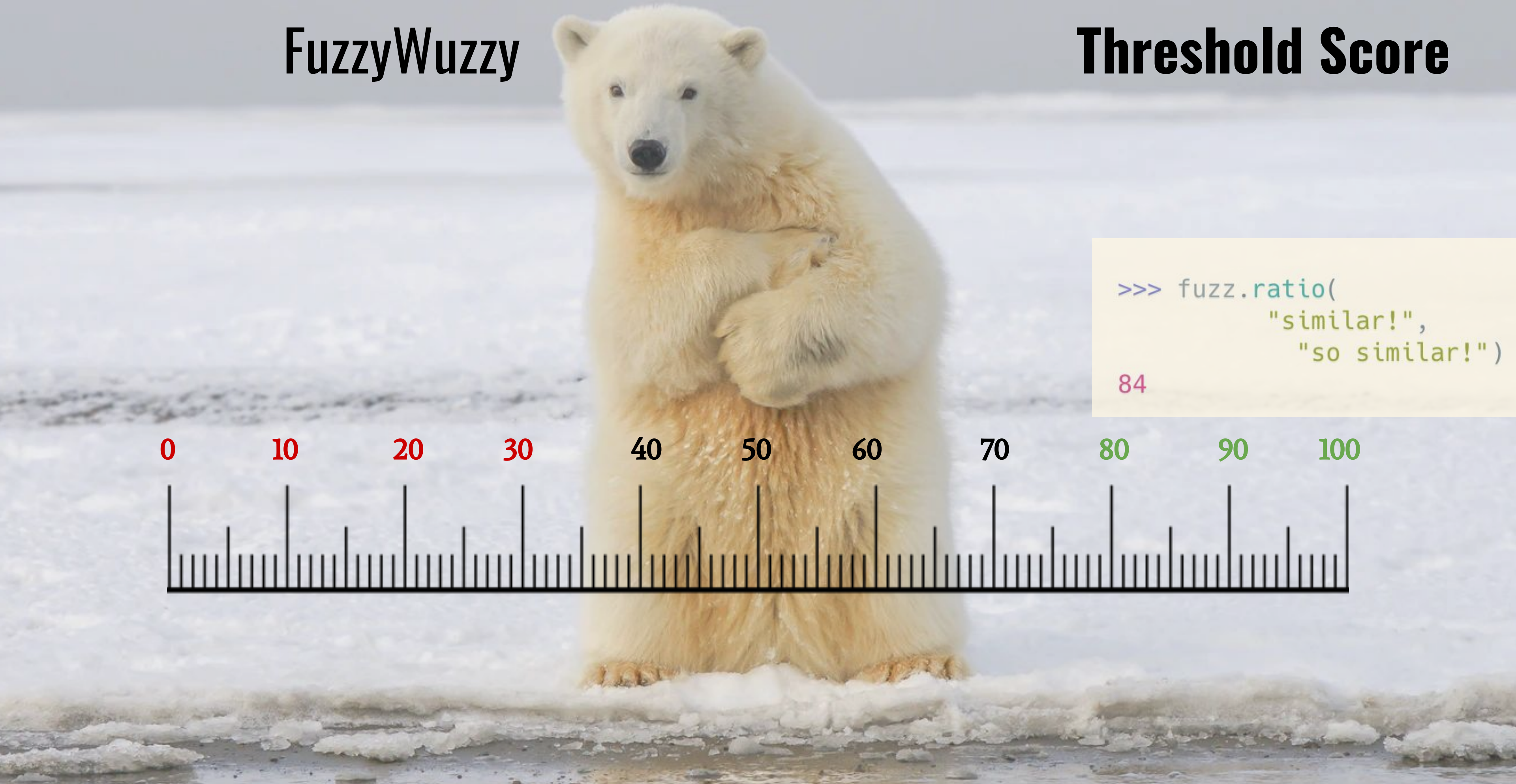


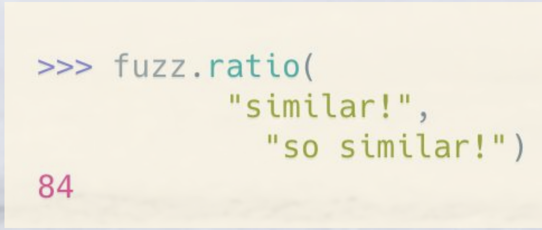
```
>>> fuzz.ratio(
            "similar!",
              "so similar!")
84
```

Naomi Kriger

FuzzyWuzzy

Threshold Score

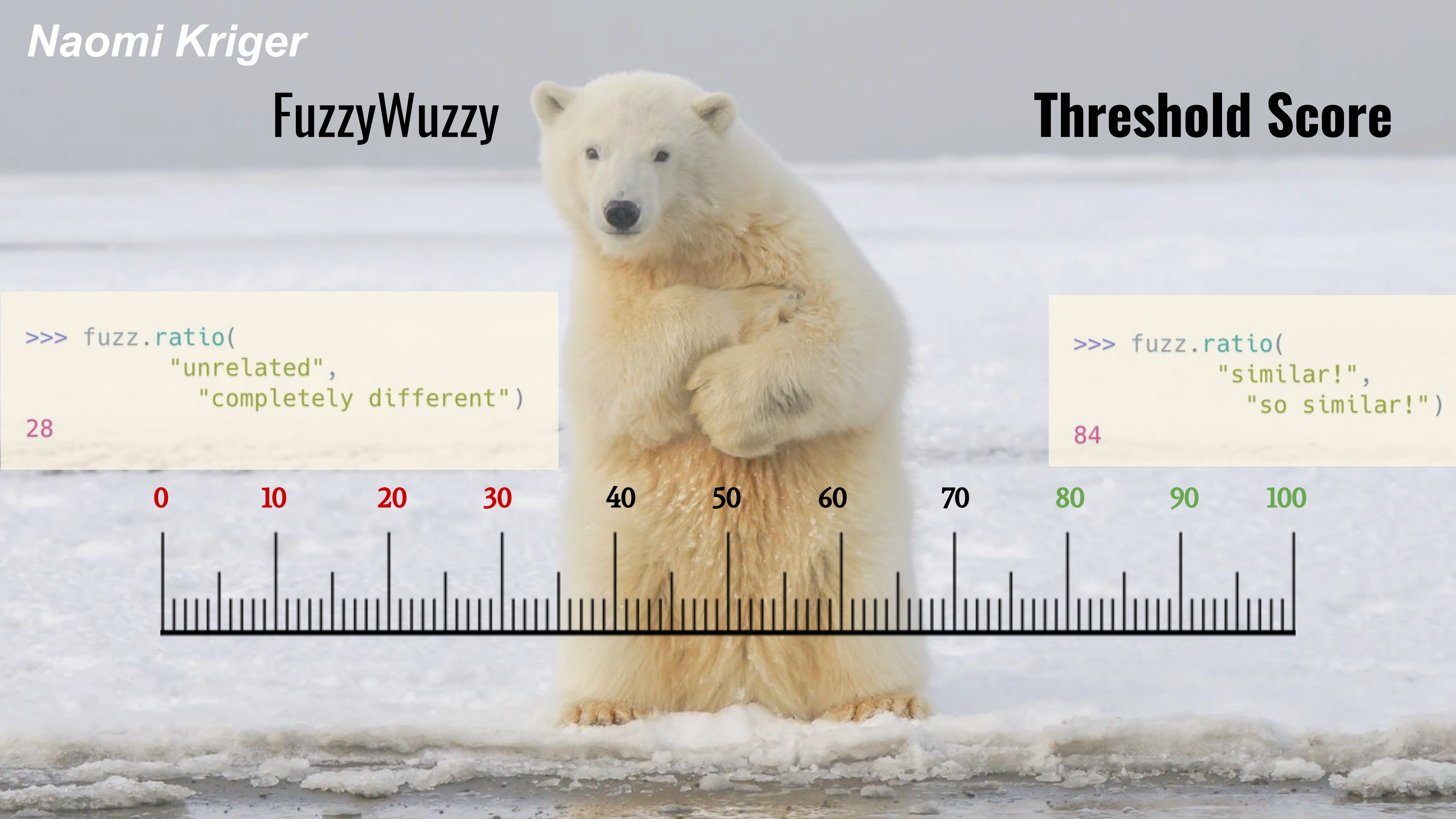




```
>>> fuzz.ratio(
          "similar!",
            "so similar!")
84
```

Naomi Kriger

FuzzyWuzzy Threshold Score

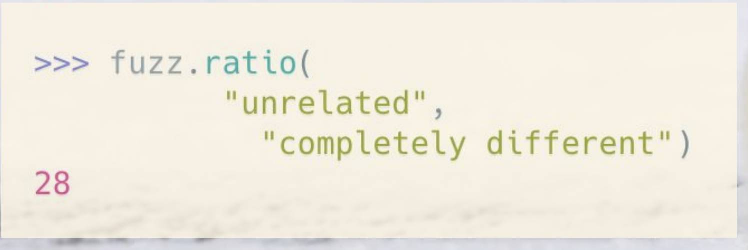


```
>>> fuzz.ratio(
        "unrelated",
          "completely different")
28
```

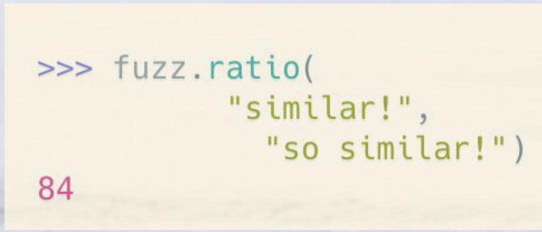
```
>>> fuzz.ratio(
          "similar!",
            "so similar!")
84
```

Naomi Kriger

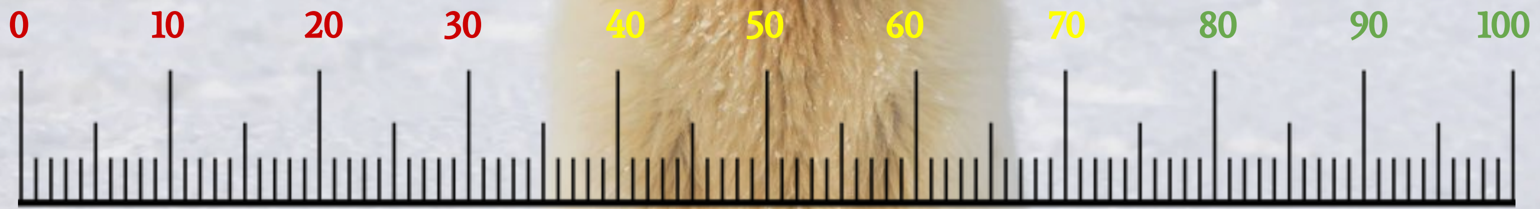
FuzzyWuzzy Threshold Score



```
>>> fuzz.ratio(
          "unrelated",
            "completely different")
28
```

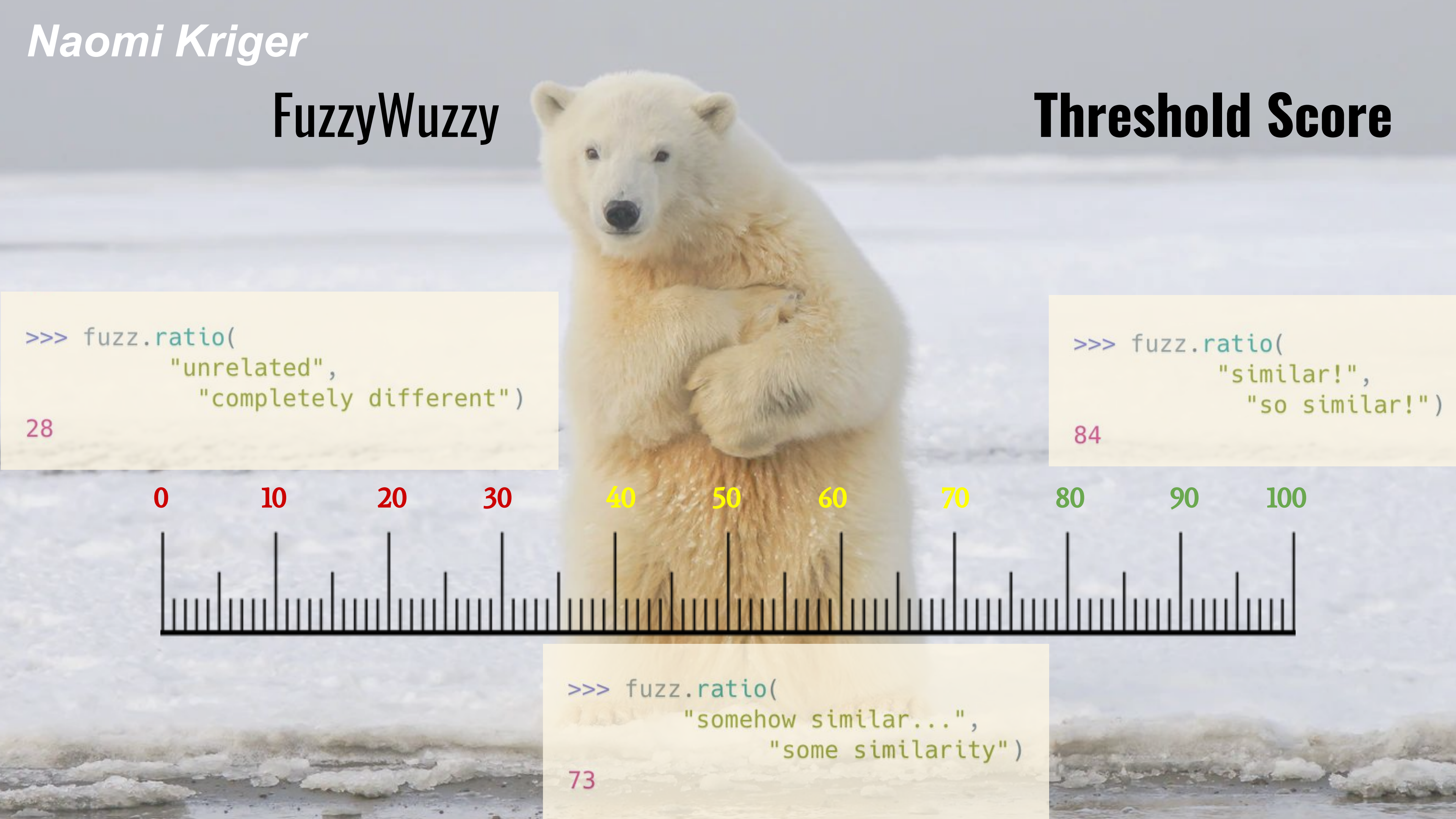


```
>>> fuzz.ratio(
          "similar!",
            "so similar!")
84
```



Naomi Kriger

FuzzyWuzzy Threshold Score



```
>>> fuzz.ratio(
        "unrelated",
          "completely different")
28
```

```
>>> fuzz.ratio(
          "similar!",
            "so similar!")
84
```

```
>>> fuzz.ratio(
        "somehow similar...",
              "some similarity")
73
```

Naomi Kriger

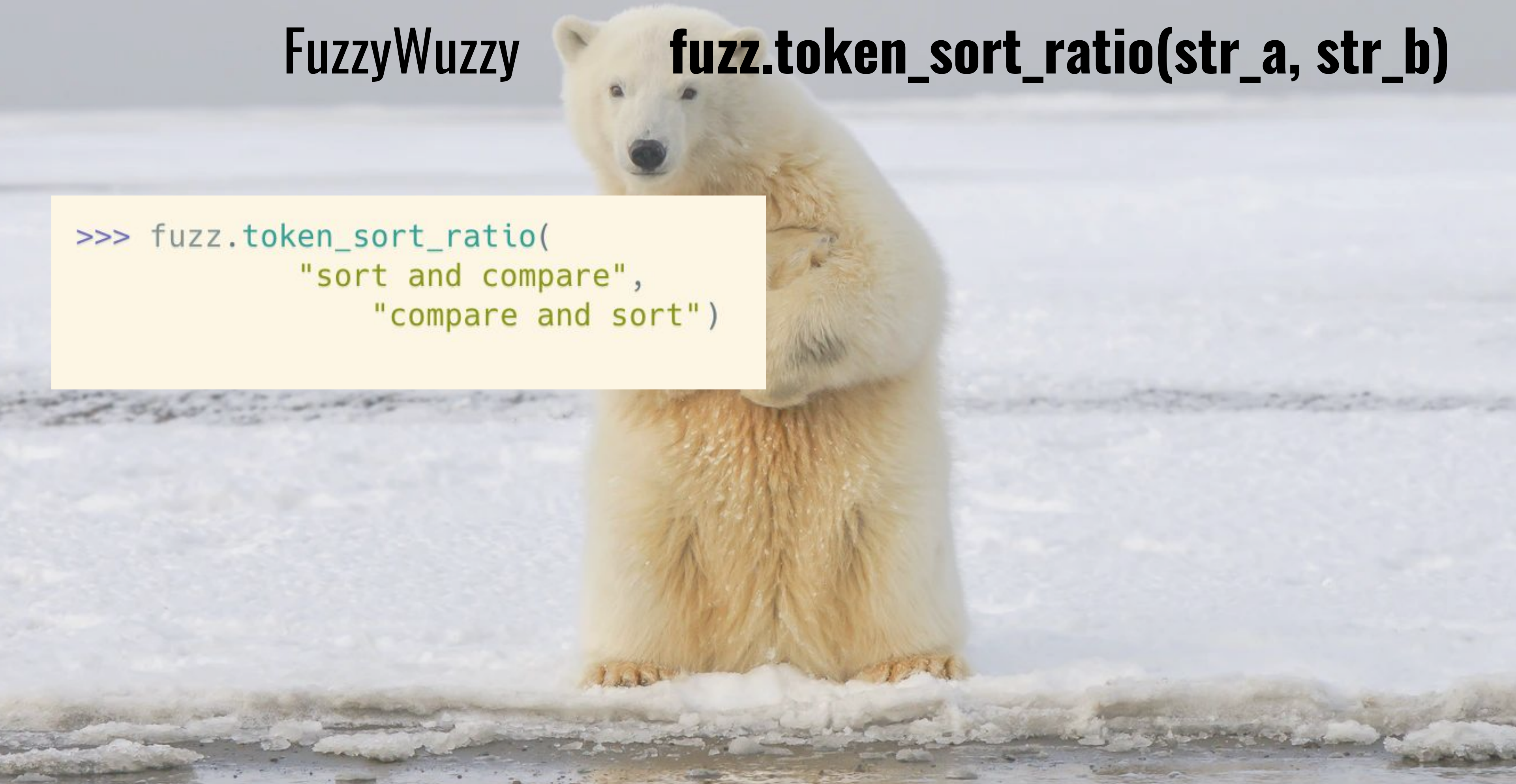
FuzzyWuzzy Threshold Score

Naomi Kriger

FuzzyWuzzy **fuzz.token_sort_ratio(str_a, str_b)**

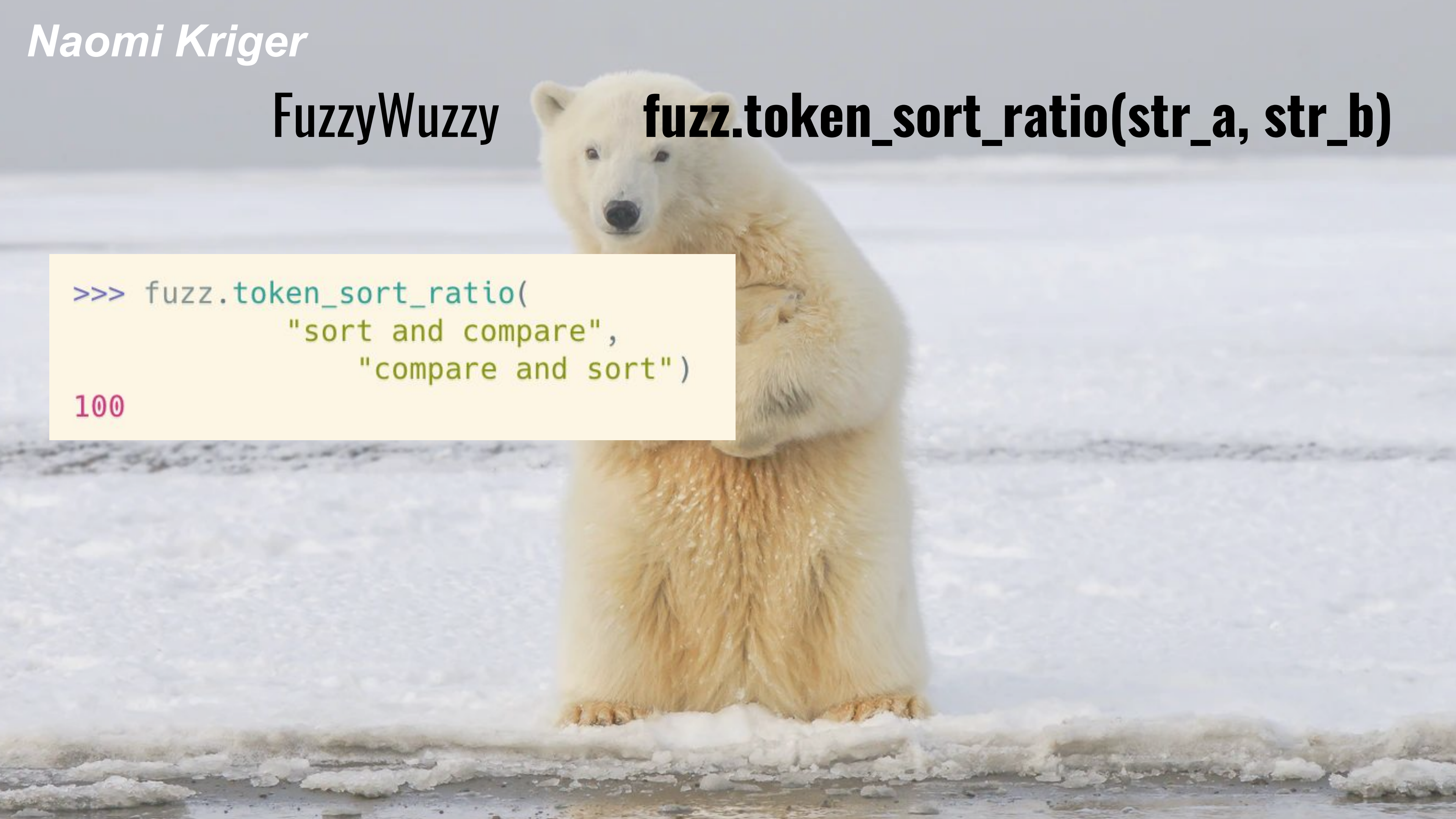
Naomi Kriger

FuzzyWuzzy fuzz.token_sort_ratio(str_a, str_b)



```
>>> fuzz.token_sort_ratio(
            "sort and compare",
                "compare and sort")
```

FuzzyWuzzy **fuzz.token_sort_ratio(str_a, str_b)**



```
>>> fuzz.token_sort_ratio(
            "sort and compare",
                "compare and sort")
100
```

Naomi Kriger

FuzzyWuzzy **fuzz.token_sort_ratio(str_a, str_b)**

```
>>> fuzz.token_sort_ratio(
                "Emma walked home with Liza",
                    "LIZA WALKED WITH EMMA HOME")
```

Naomi Kriger

FuzzyWuzzy **fuzz.token_sort_ratio(str_a, str_b)**

```
>>> fuzz.token_sort_ratio(
            "Emma walked home with Liza",
                "LIZA WALKED WITH EMMA HOME")
100
```

Naomi Kriger

FuzzyWuzzy

`fuzz.token_set_ratio(str_a, str_b)`

FuzzyWuzzy fuzz.token_set_ratio(str_a, str_b)

```
>>> fuzz.token_set_ratio(
            "sort, lower, and no-repeats",
                "LOWER, no-repeats, sort and sort")
```

FuzzyWuzzy fuzz.token_set_ratio(str_a, str_b)

```
>>> fuzz.token_set_ratio(
            "sort, lower, and no-repeats",
                "LOWER, no-repeats, sort and sort")
100
```

FuzzyWuzzy fuzz.token_set_ratio(str_a, str_b)

```
>>> fuzz.token_set_ratio(
              "sort, lower, and no-repeats",
                   "LOWER, no-repeats, sort and sort")
100

>>> fuzz.token_sort_ratio(
              "sort, lower, and no-repeats",
                   "LOWER, no-repeats, sort and sort")
```

FuzzyWuzzy

fuzz.token_set_ratio(str_a, str_b)

```
>>> fuzz.token_set_ratio(
            "sort, lower, and no-repeats",
                "LOWER, no-repeats, sort and sort")
100

>>> fuzz.token_sort_ratio(
            "sort, lower, and no-repeats",
                "LOWER, no-repeats, sort and sort")
91
```

FuzzyWuzzy **fuzz.token_set_ratio(str_a, str_b)**

```
>>> fuzz.token_set_ratio(
             "I love chocolate and ice cream",
                 "I LOVE ice cream AND I love chocolate!")
```

Naomi Kriger

FuzzyWuzzy fuzz.token_set_ratio(str_a, str_b)

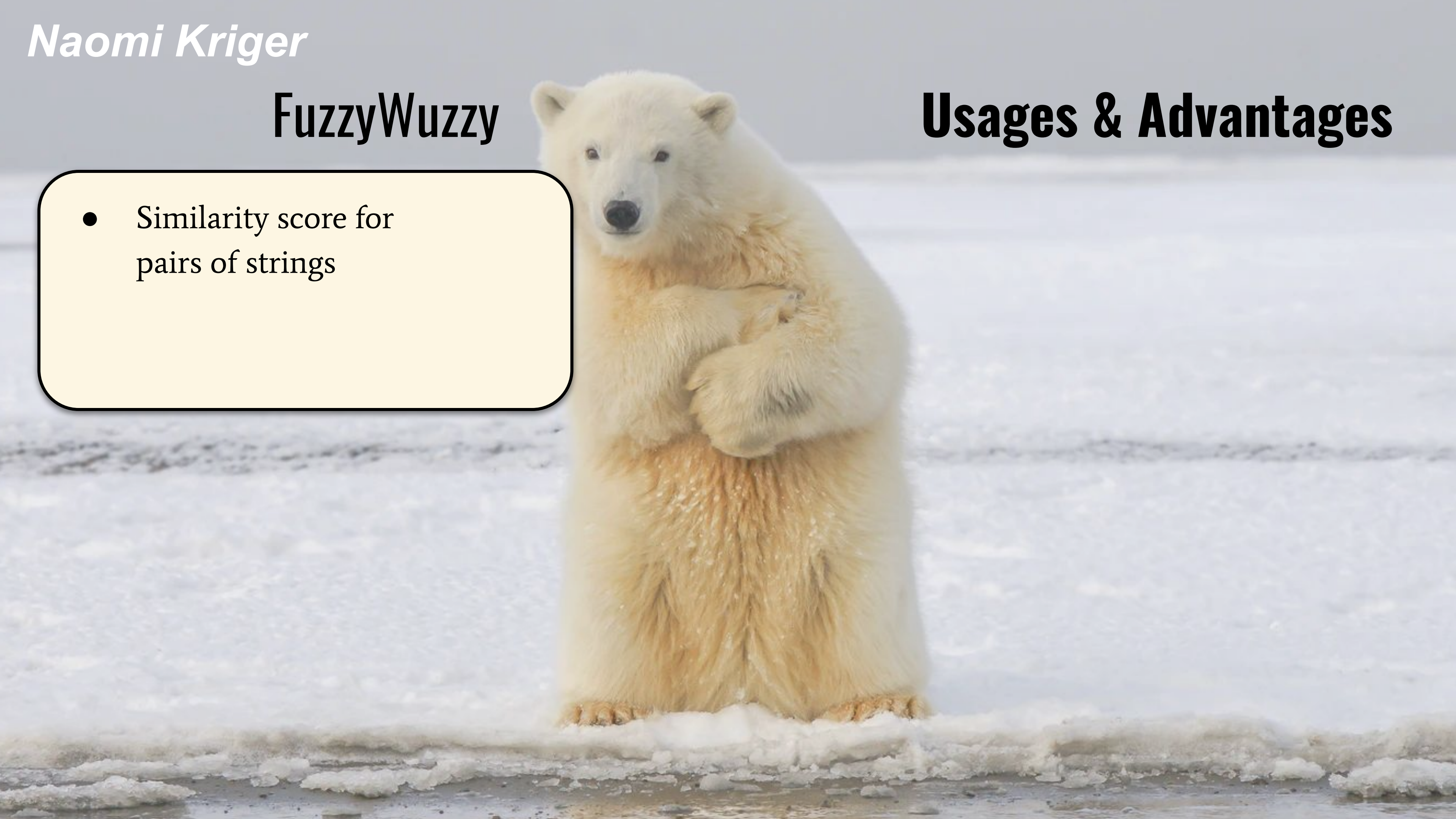
```
>>> fuzz.token_set_ratio(
            "I love chocolate and ice cream",
                "I LOVE ice cream AND I love chocolate!")
100
```

Naomi Kriger

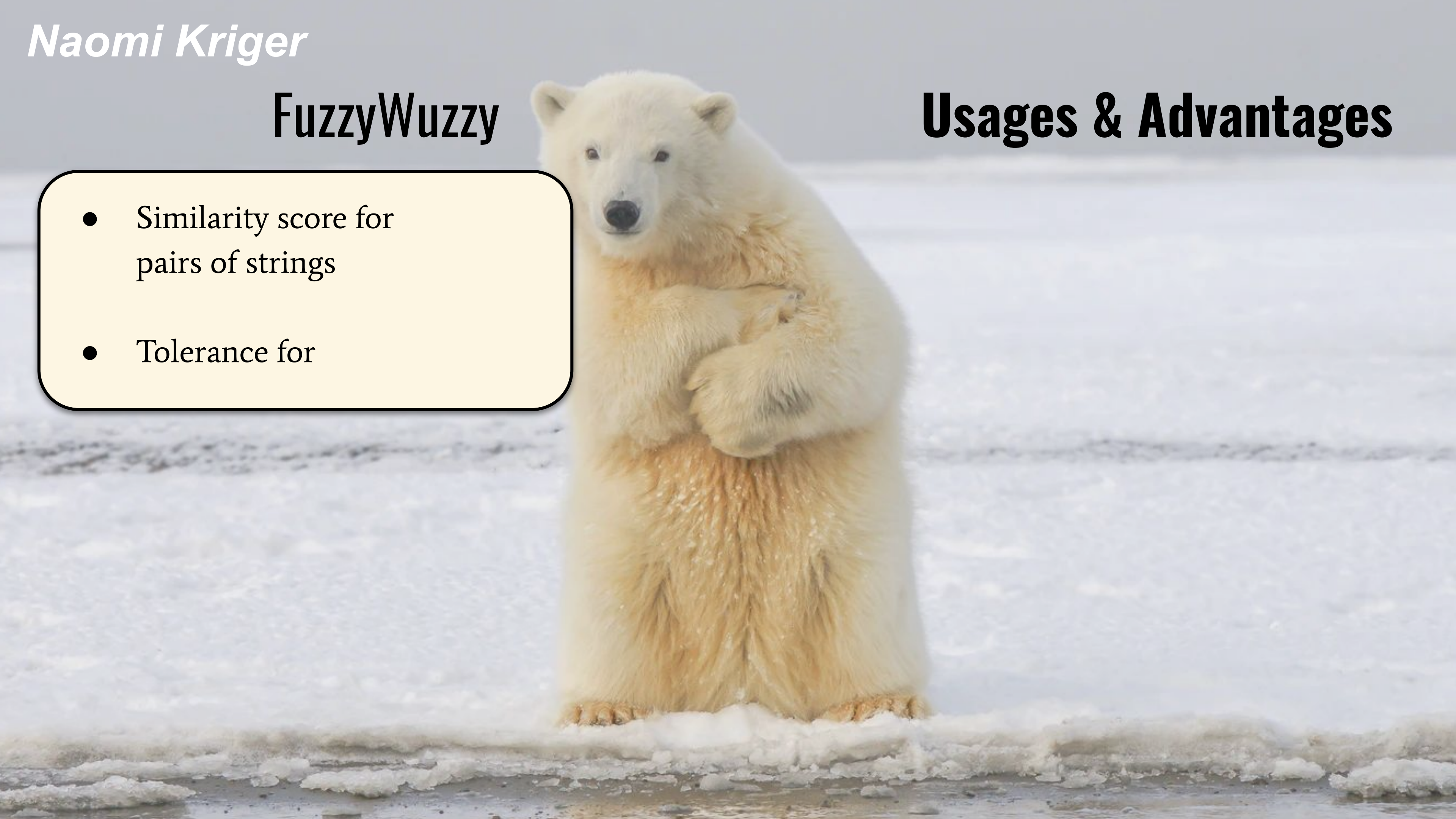
FuzzyWuzzy Usages & Advantages

Naomi Kriger

FuzzyWuzzy Usages & Advantages



- Similarity score for pairs of strings

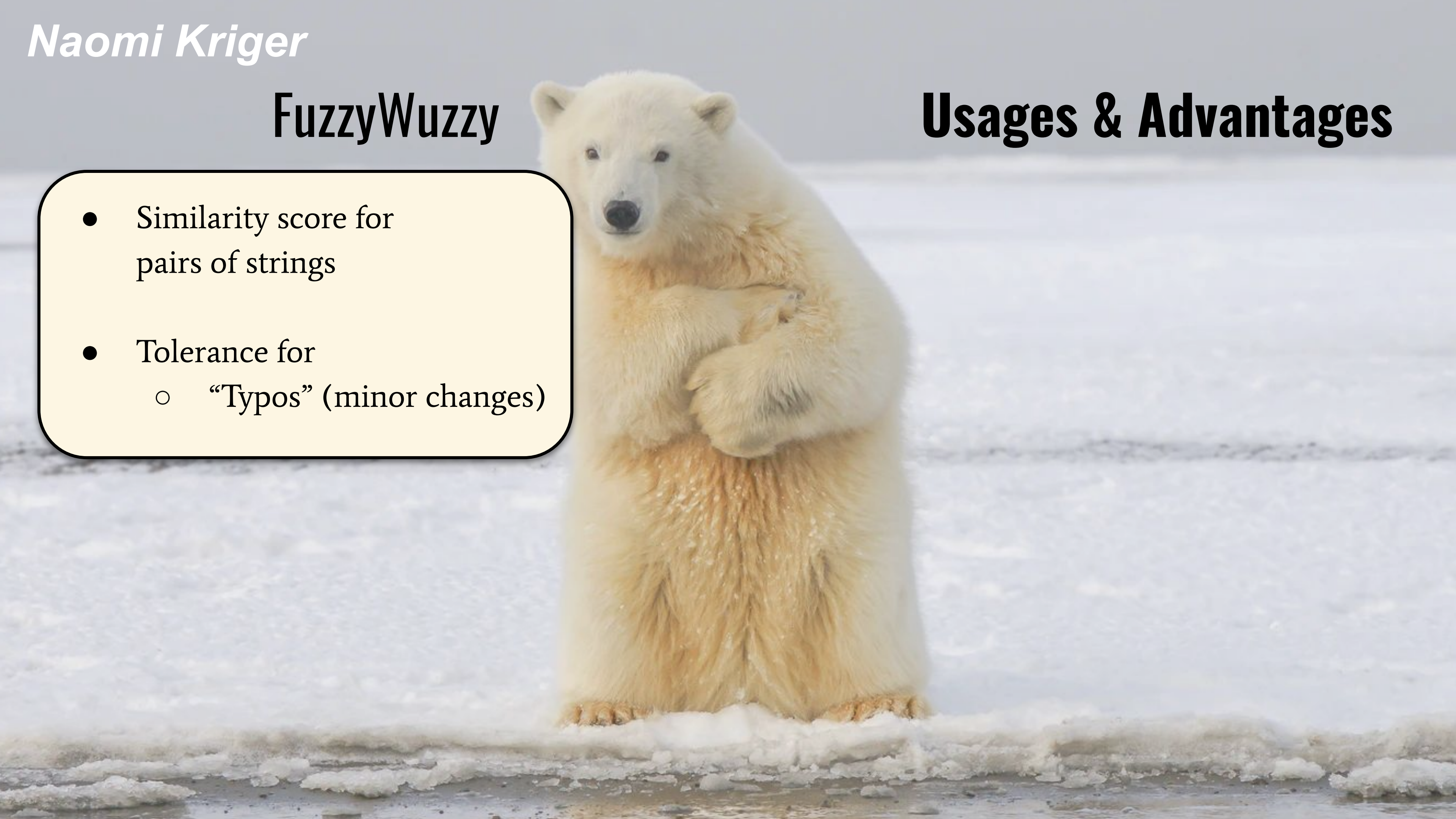


Naomi Kriger

FuzzyWuzzy

Usages & Advantages

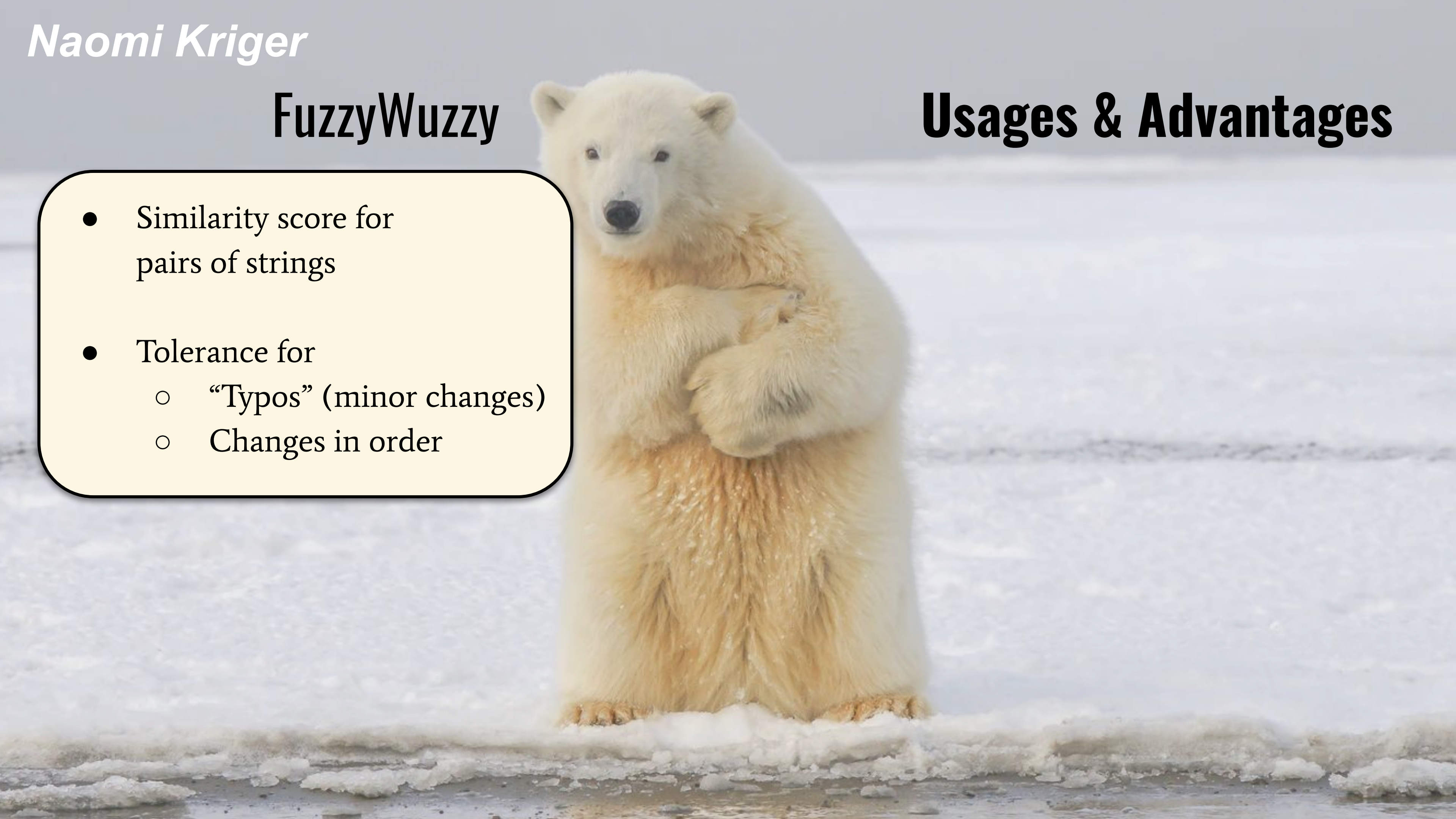
- Similarity score for pairs of strings
- Tolerance for



FuzzyWuzzy Usages & Advantages

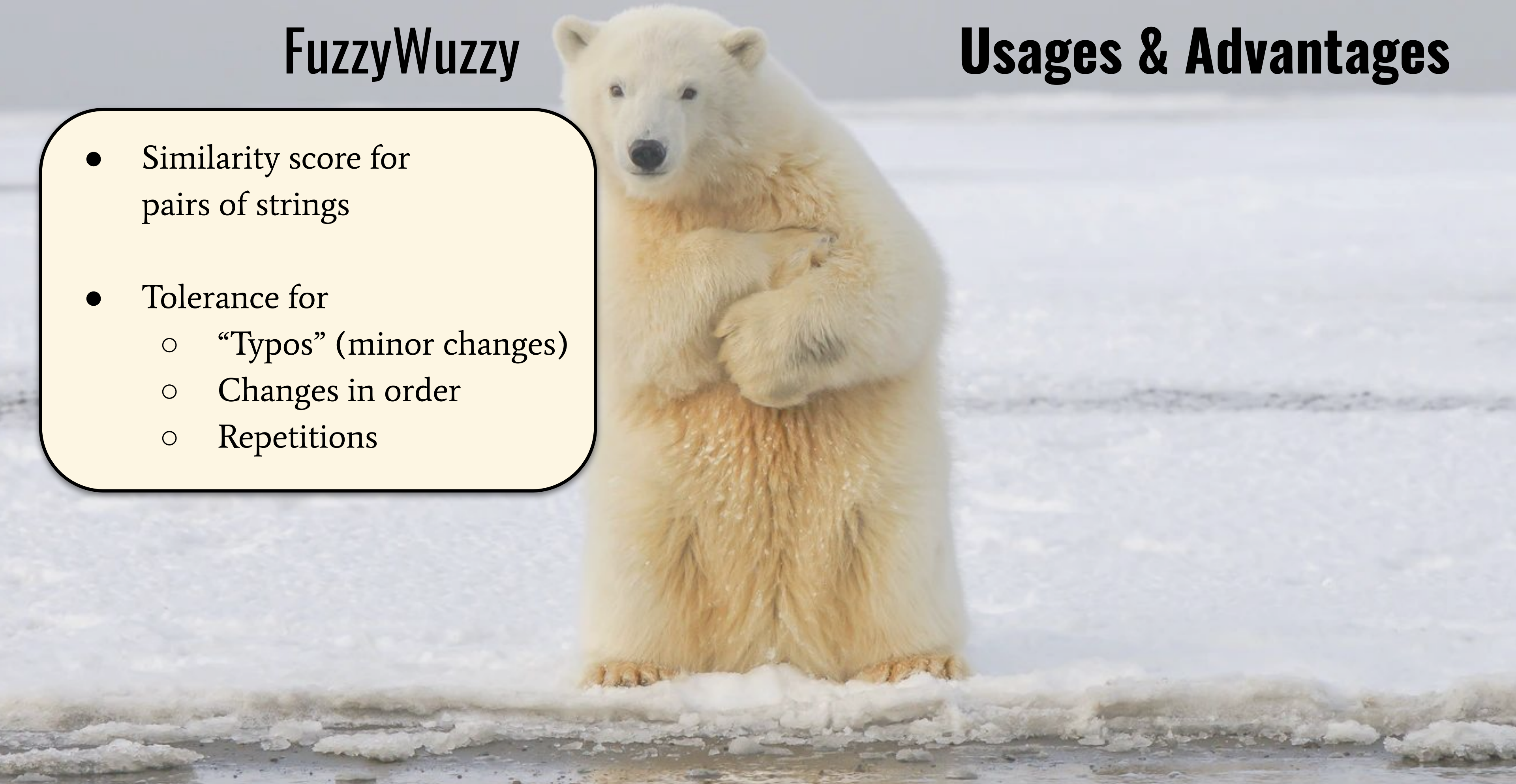
- Similarity score for pairs of strings
- Tolerance for
  - "Typos" (minor changes)

FuzzyWuzzy Usages & Advantages



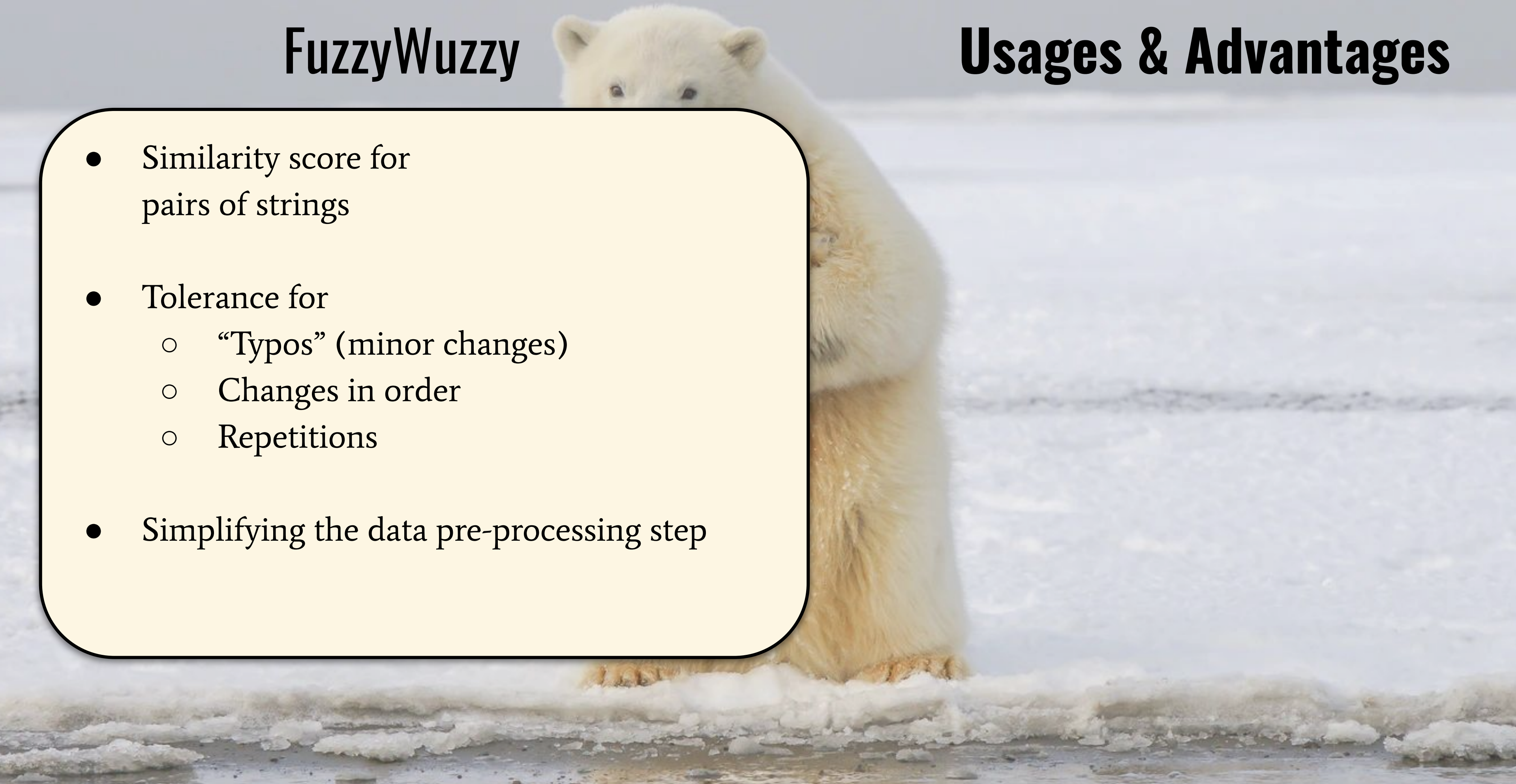
- Similarity score for pairs of strings
- Tolerance for
    - “Typos” (minor changes)
    - Changes in order

FuzzyWuzzy Usages & Advantages



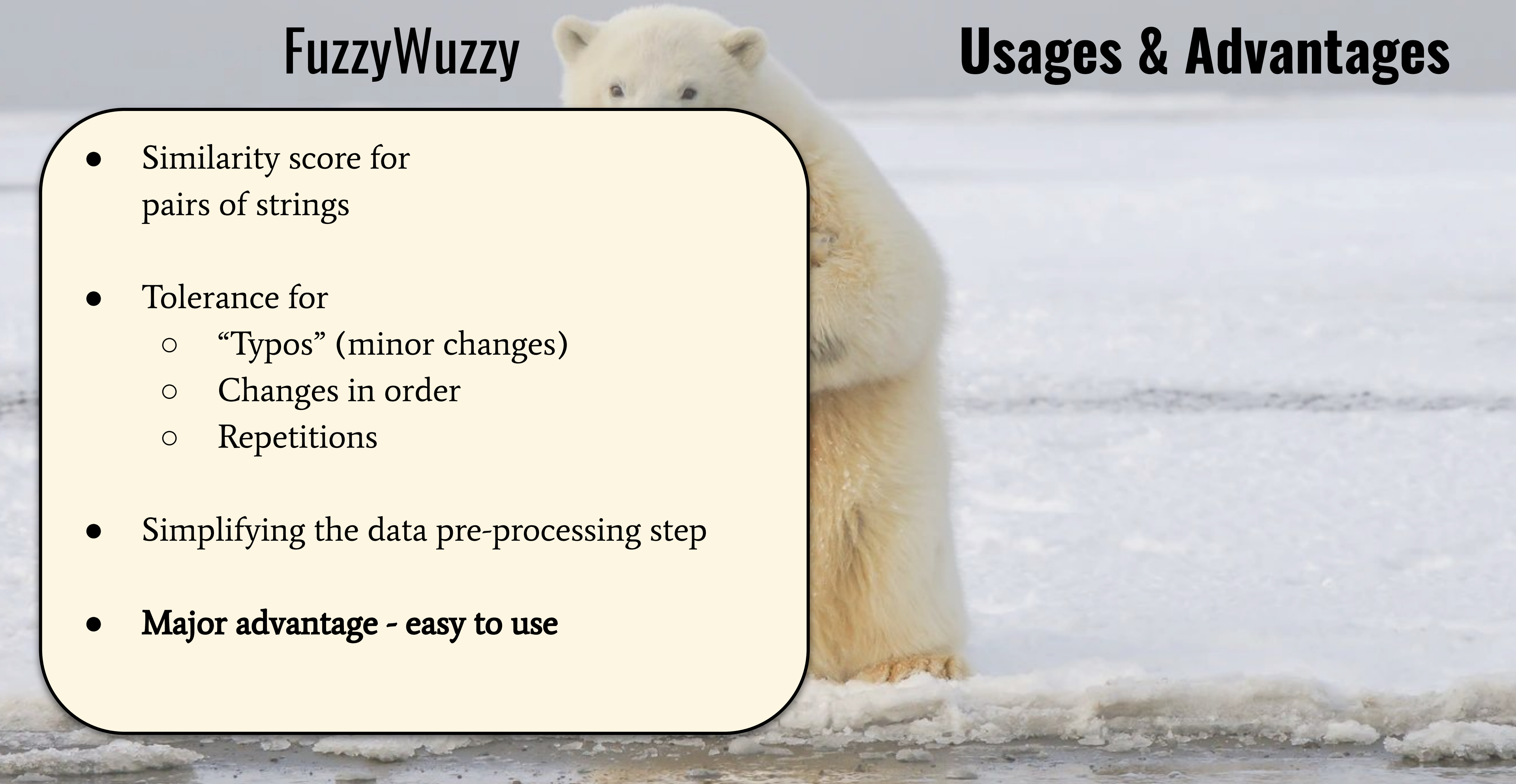
- Similarity score for pairs of strings
- Tolerance for
  - “Typos” (minor changes)
  - Changes in order
  - Repetitions

FuzzyWuzzy Usages & Advantages



- Similarity score for pairs of strings
- Tolerance for
  - “Typos” (minor changes)
  - Changes in order
  - Repetitions
- Simplifying the data pre-processing step

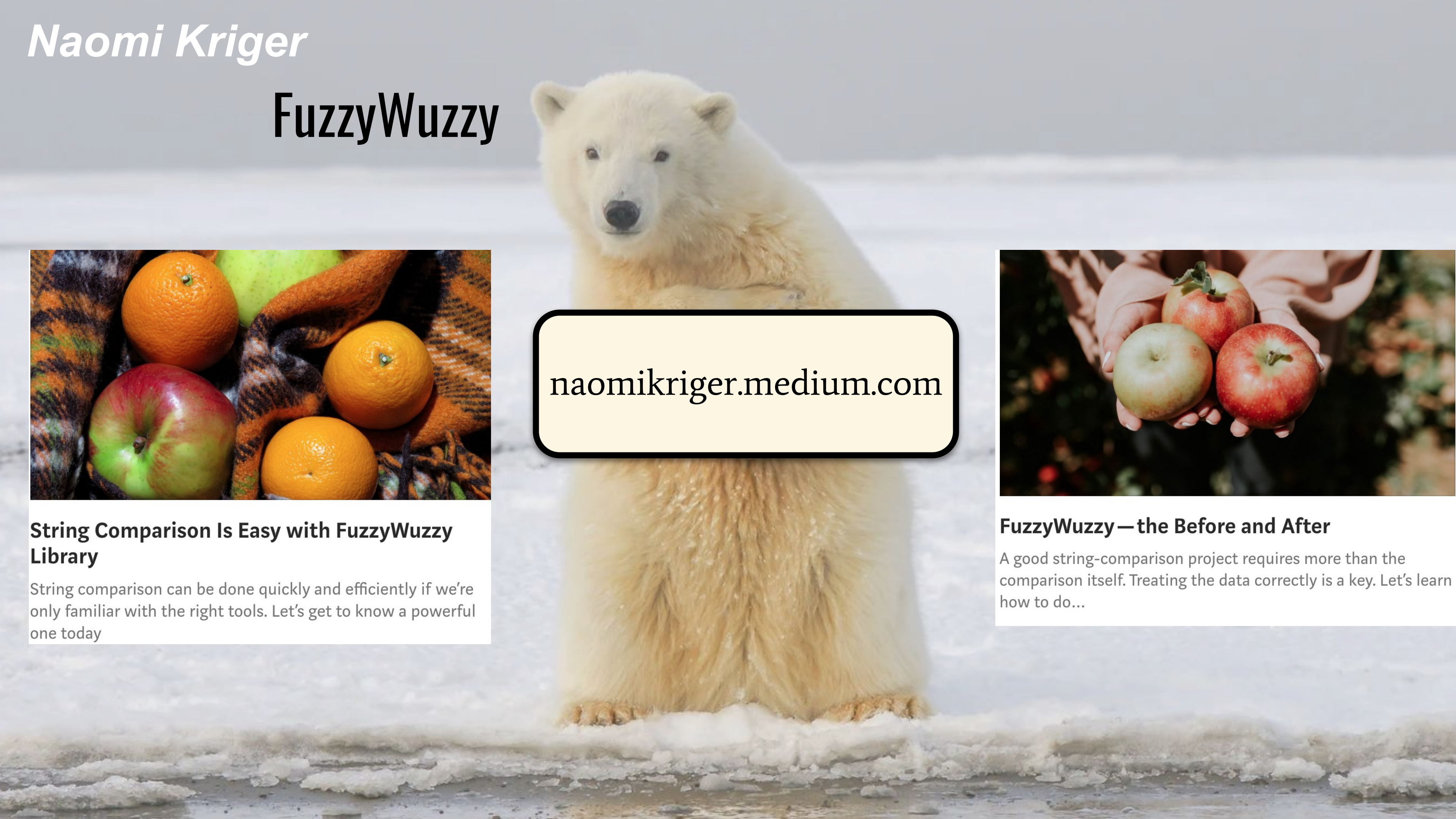
FuzzyWuzzy Usages & Advantages



- Similarity score for pairs of strings
- Tolerance for
  - “Typos” (minor changes)
  - Changes in order
  - Repetitions
- Simplifying the data pre-processing step
- **Major advantage - easy to use**

Naomi Kriger

FuzzyWuzzy



naomikriger.medium.com

String Comparison Is Easy with FuzzyWuzzy Library

String comparison can be done quickly and efficiently if we're only familiar with the right tools. Let's get to know a powerful one today

FuzzyWuzzy — the Before and After

A good string-comparison project requires more than the comparison itself. Treating the data correctly is a key. Let's learn how to do...

Naomi Kriger

What did we learn?

Naomi Kriger

What did we learn?

Jellyfish

Naomi Kriger

What did we learn?

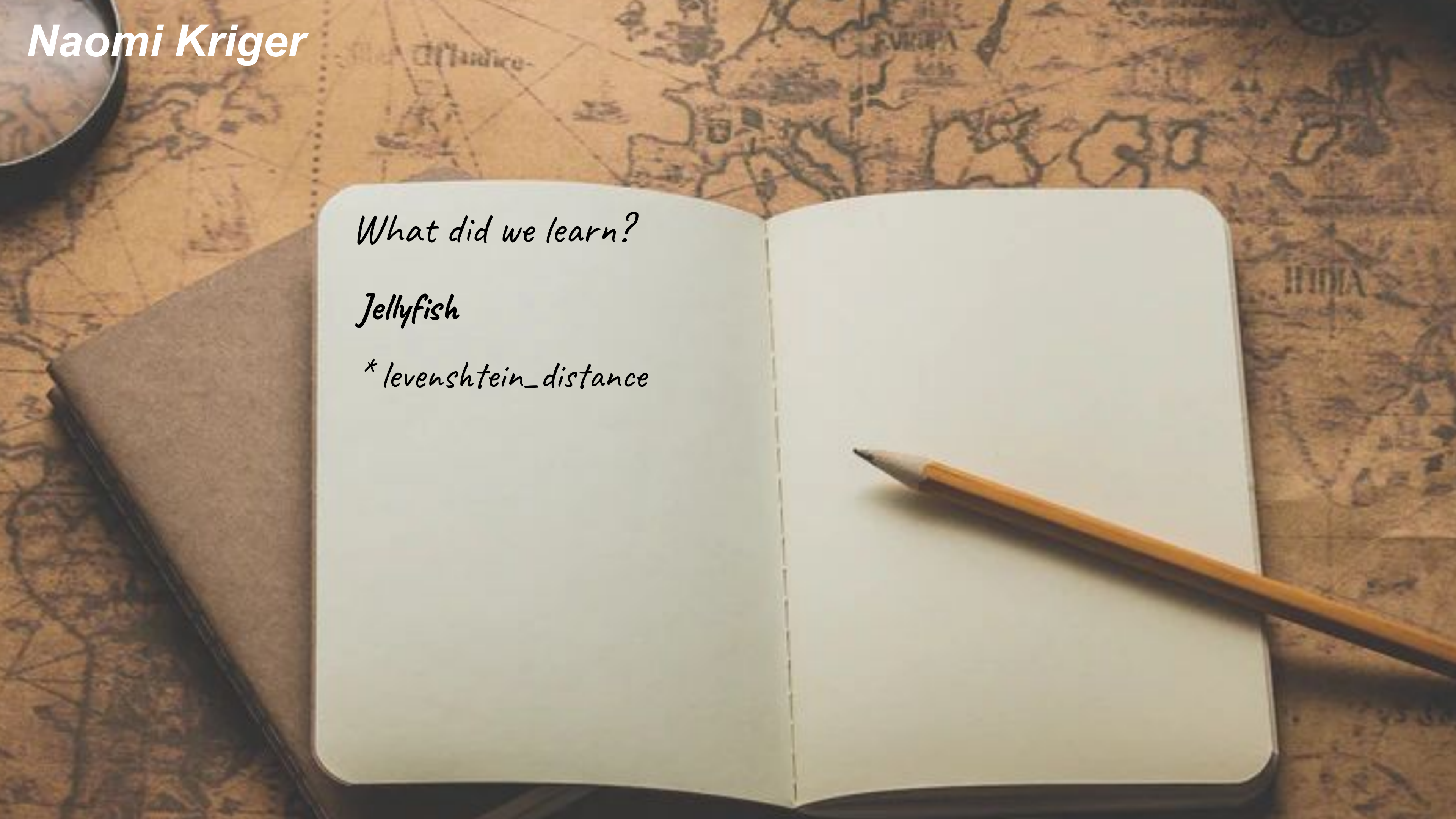
Jellyfish

FuzzyWuzzy

Naomi Kriger

What did we learn?

Jellyfish

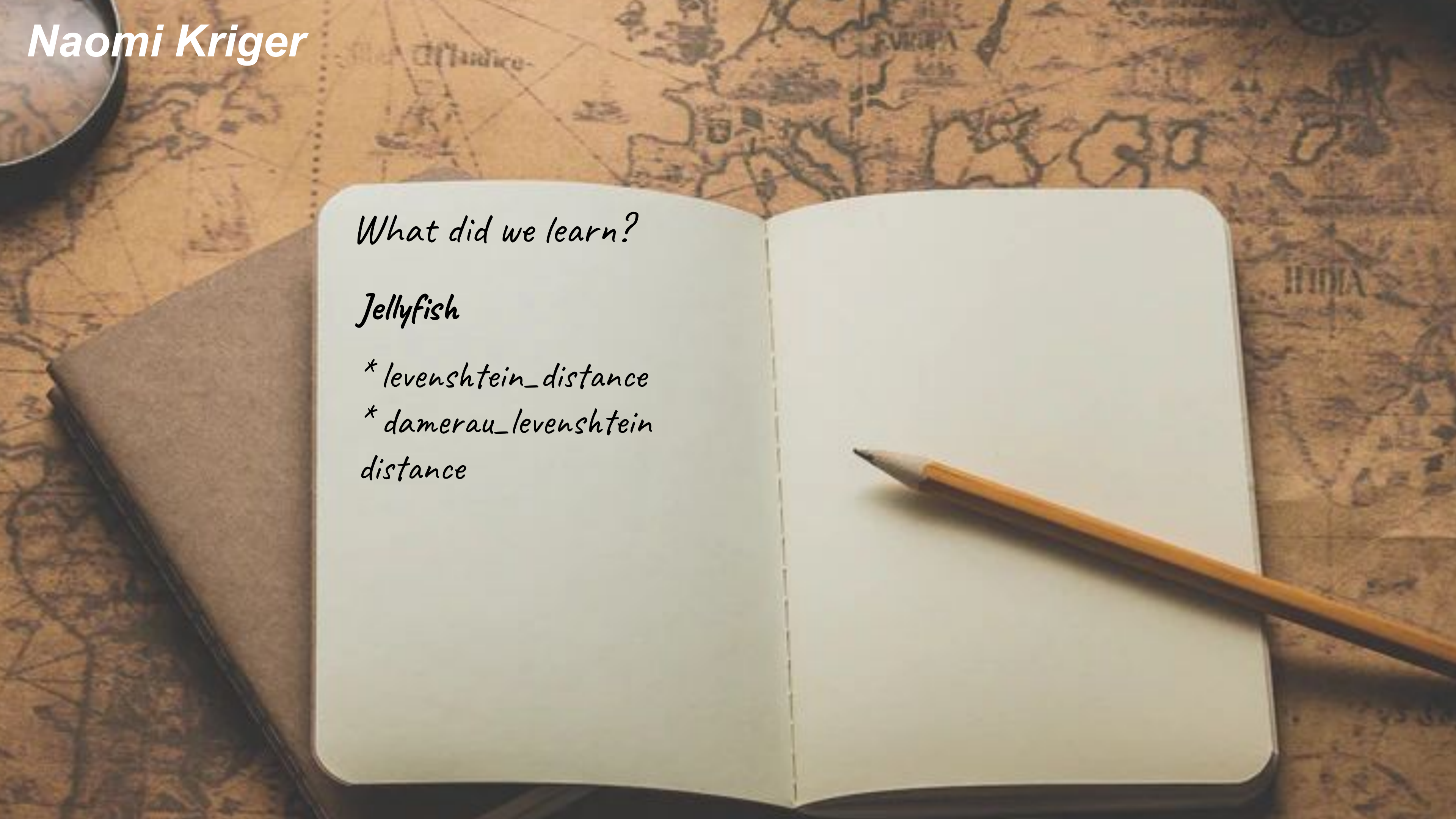


What did we learn?

Jellyfish

* levenshtein_distance

Naomi Kriger



What did we learn?

Jellyfish

* levenshtein_distance
* damerau_levenshtein distance

Naomi Kriger

What did we learn?

Jellyfish

* levenshtein_distance
* damerau_levenshtein distance

FuzzyWuzzy

Naomi Kriger

What did we learn?

Jellyfish

* levenshtein_distance
* damerau_levenshtein distance

FuzzyWuzzy

* fuzz.ratio

Naomi Kriger

What did we learn?

Jellyfish

* levenshtein_distance
* damerau_levenshtein distance

FuzzyWuzzy

* fuzz.ratio
* fuzz.token_sort_ratio

What did we learn?

Jellyfish

* levenshtein_distance
* damerau_levenshtein distance

FuzzyWuzzy

* fuzz.ratio
* fuzz.token_sort_ratio
* fuzz.token_set_ratio

linkedin.com/in/naomi-kriger

naomikriger.medium.com

thanks!
Tombow ABT
Acid Free
N15

String Comparison In Real Life

Naomi Kriger