

# Normal Form Grapheme



Jonathan Worthington



Hi!  
I'm still Jonathan.



# unicode

SHOP NO 1  
ABBA'S  
DEPT. STORE  
MADE TO MEASURE  
CLOTHES H. 12 HRS  
TAILORS

牛車水  
食街

Chinatown  
Food Street

敦煌劇坊戲曲票館

新嘉坡牛社

新聲詩社

PIZZERIA

CHINATOWN FOOD STREET  
Mediterranean, Indian, Italian, Chinese

233

聚點

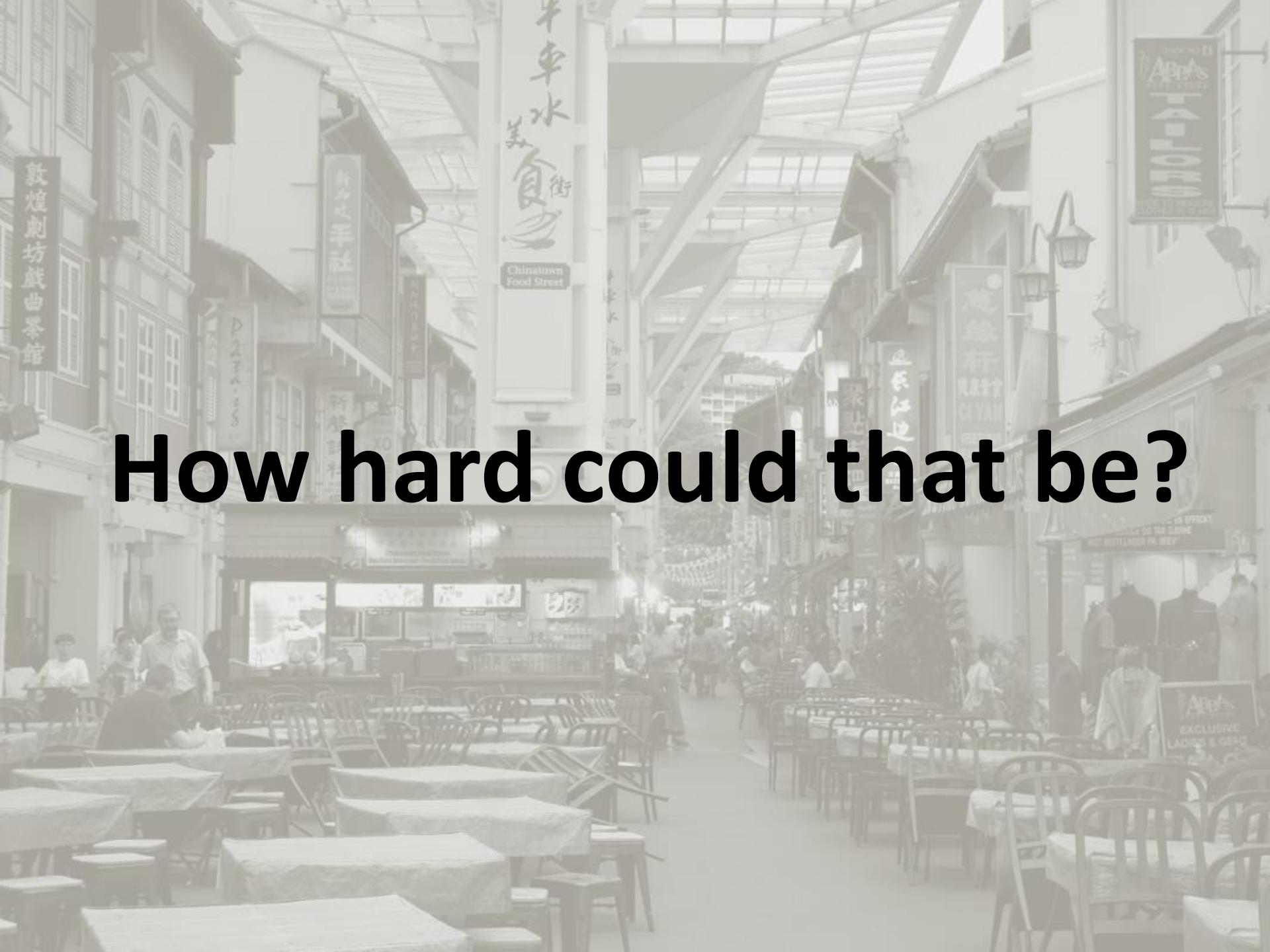
長江連

慈緣軒  
健康素食  
CI YAN  
VEGETARIAN  
HEALTHY

DR. EFFEKT  
MOT BESTILLINGER PÅ BREV

ABBA'S  
DEPT. STORE  
EXCLUSIVE  
LADIES & GENT

A unified scheme to  
encode and manipulate  
all the writing systems of  
the world (present,  
historical, math, emoji...)



# How hard could that be?



Well, let's look at some  
languages.

# English

Aardvarks don't make a tasty meal.

Just give numbers to each character in the alphabet. Easy!

# Russian

Мне нравится пиво!

Just another alphabet. More numbers. Easy!

# Chinese?

你好吗？ 很好，谢谢。

A character set. Thousands and thousands more numbers.

# Slovak

**Slovenky sú veľmi pekné.**

**Doesn't seem too scary. I mean,  
it's Latin-based. But...**

# Slovak

**Slovenky sú veľmi pekné.**

...what about the diacritics? Do we give numbers to them?

# Slovak

Slovenky sú veľmi pekné.

Or number the combinations of letters with diacritics?



# Slovak

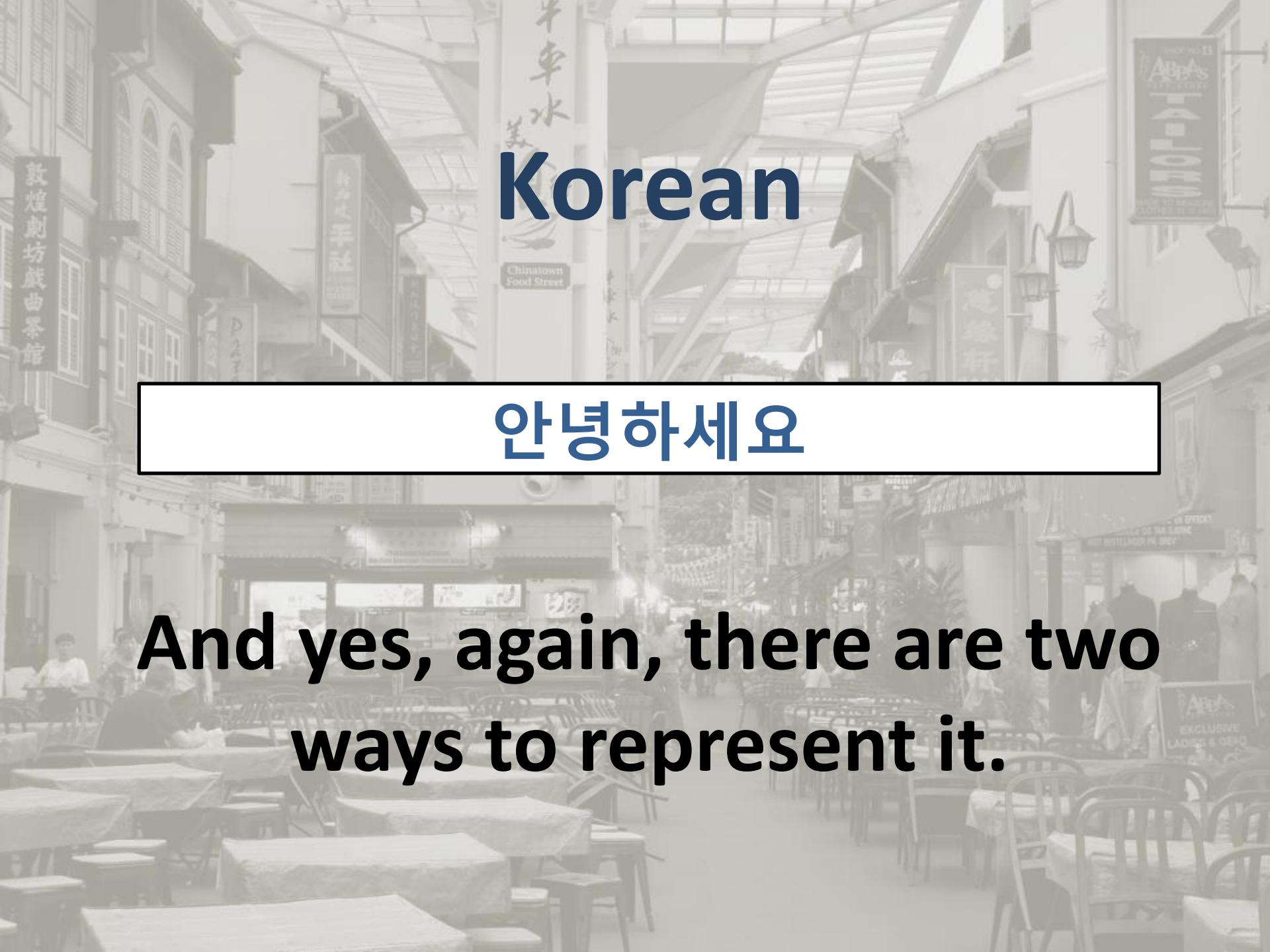
Slovenky sú veľmi pekné.

Spoiler: it's Unicode, so of course we...do *both!*

# Korean

안녕하세요

A syllabary, with each syllable block made of 2-3 "atoms".



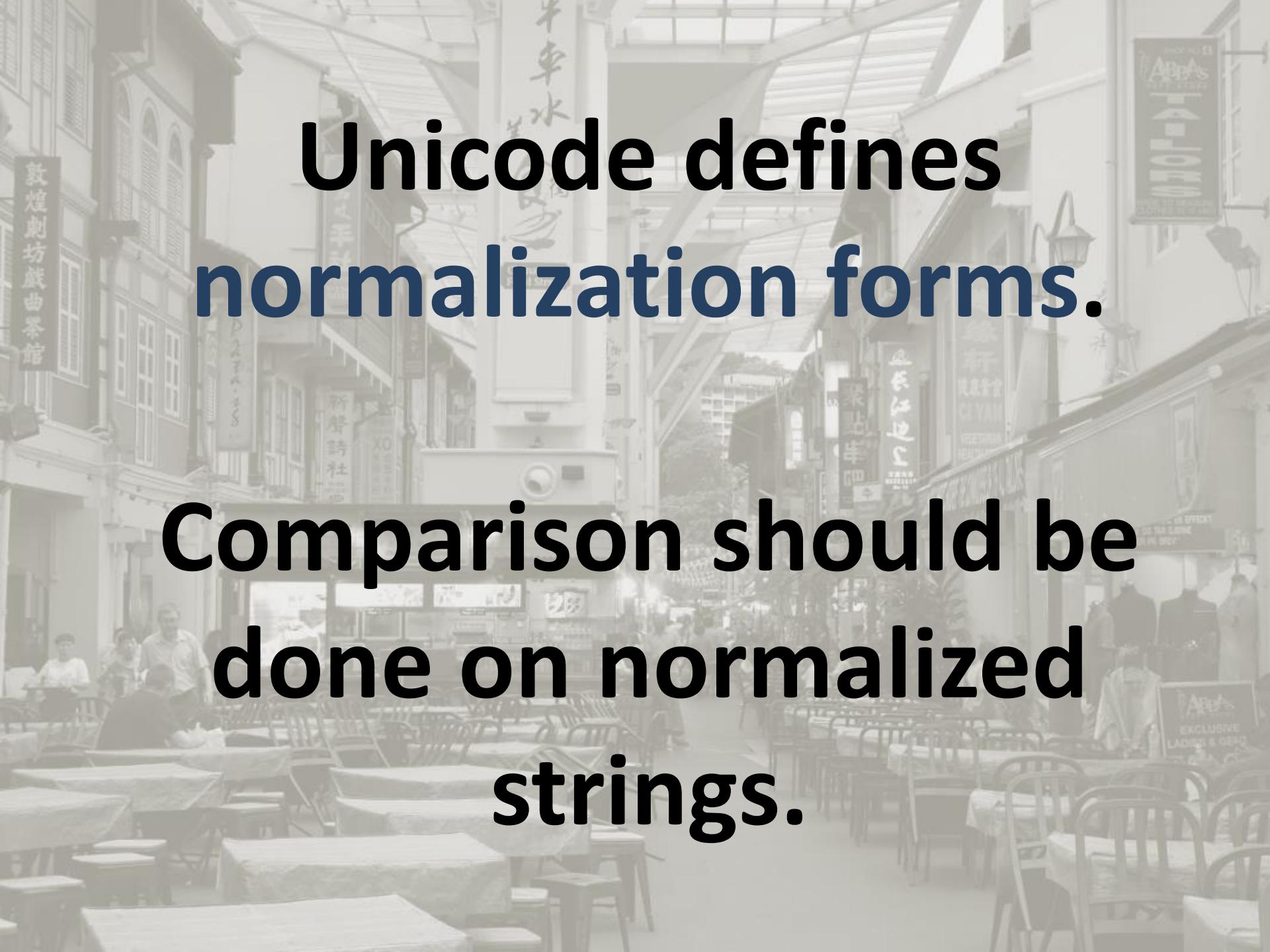
# Korean

안녕하세요

And yes, again, there are two ways to represent it.

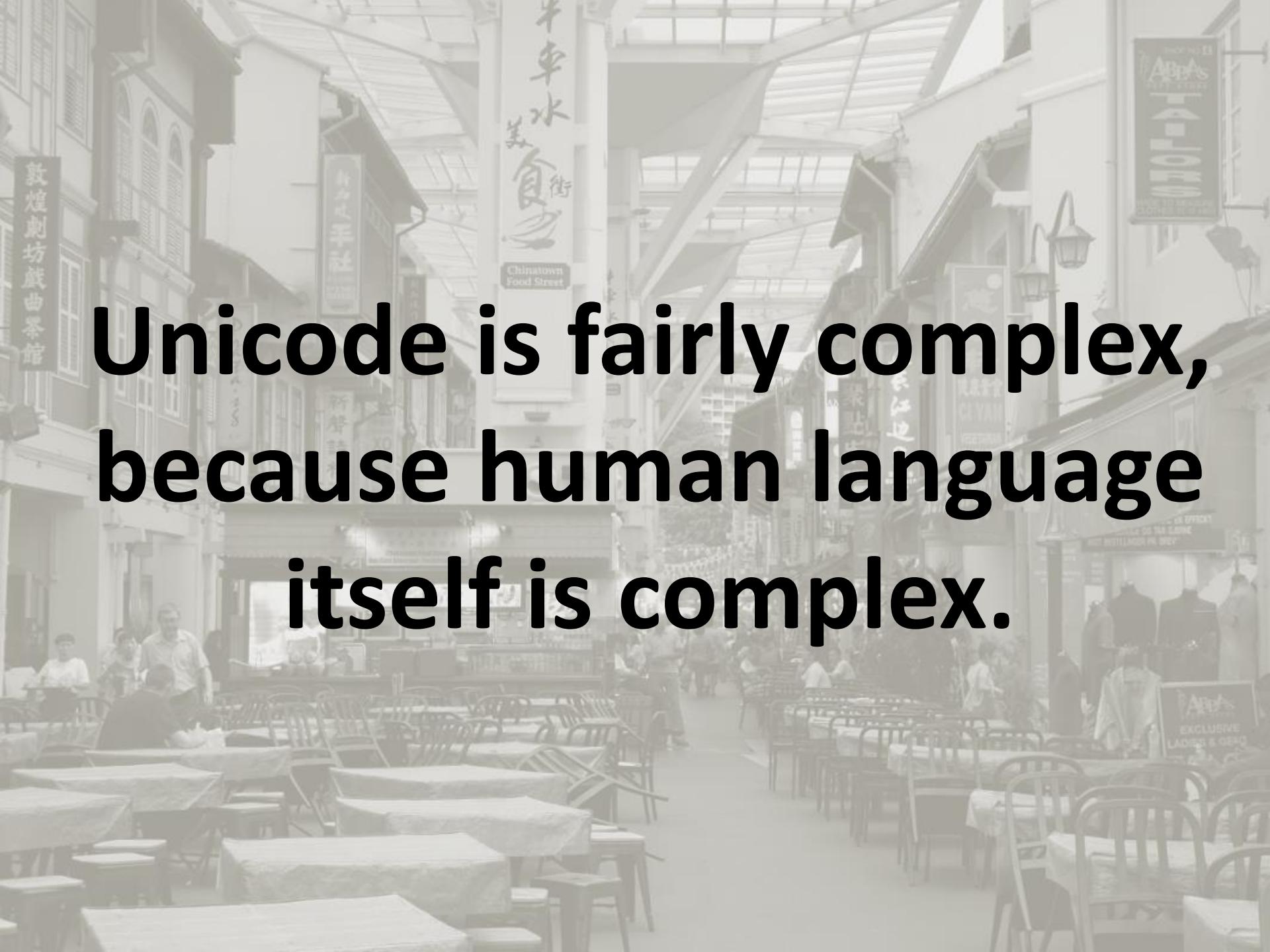


So how do we even  
compare strings?!

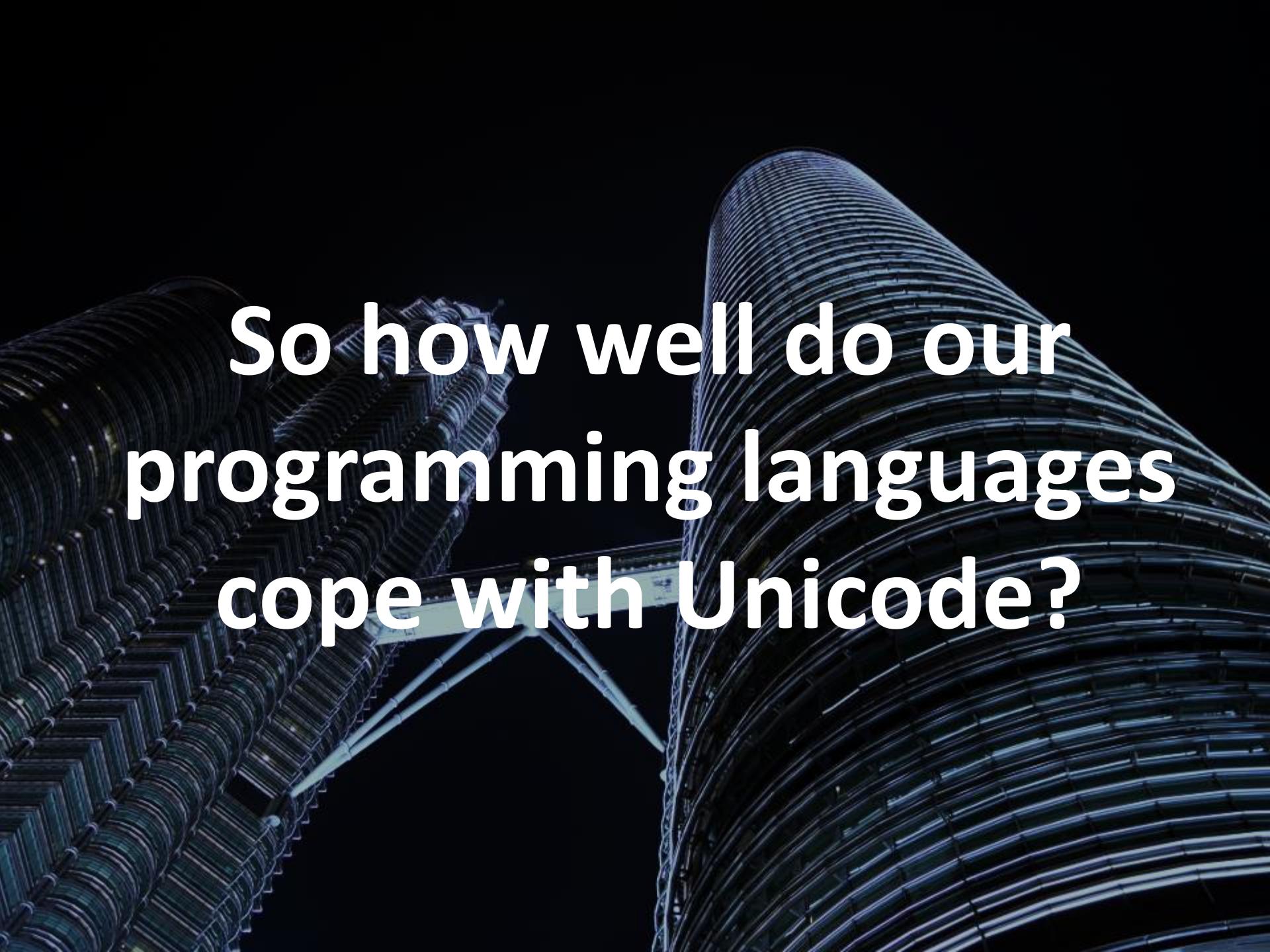


Unicode defines  
normalization forms.

Comparison should be  
done on normalized  
strings.

A grayscale photograph of a busy Chinatown street. In the foreground, there are several tables and chairs set up outdoors. People are walking around, some sitting at the tables. The background is filled with buildings, many of which have vertical signs with Chinese characters. One prominent sign on the left reads "敦煌刺坊戲曲琴館". Another sign on the right says "Chinatown Food Street".

**Unicode is fairly complex,  
because human language  
itself is complex.**

The background of the slide features a dark, abstract design composed of numerous thin, glowing blue lines that curve and flow across the frame, creating a sense of depth and motion.

So how well do our  
programming languages  
cope with Unicode?

U+1F639

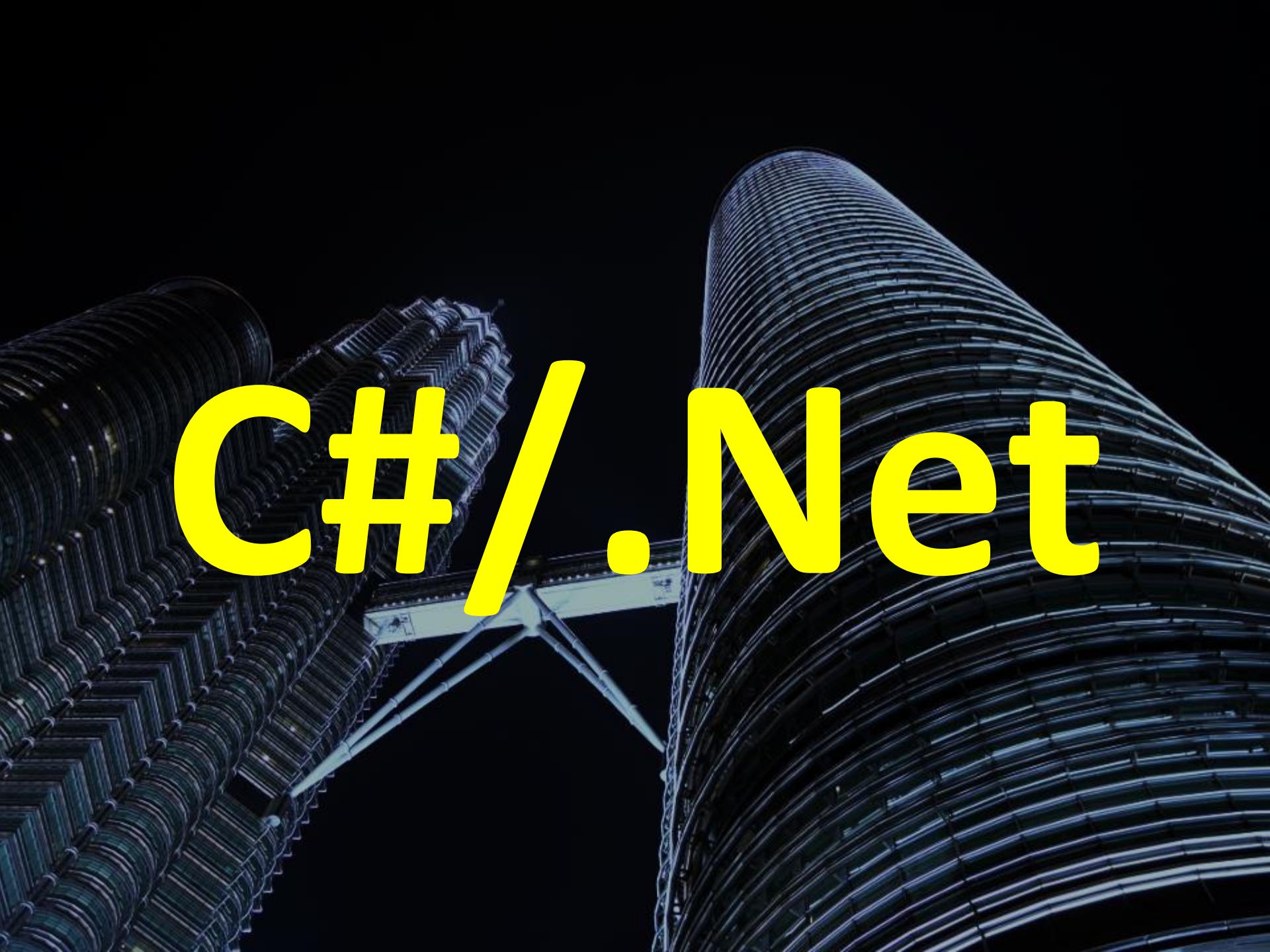


CAT FACE WITH TEARS OF JOY

U+0044 U+0323 U+0307

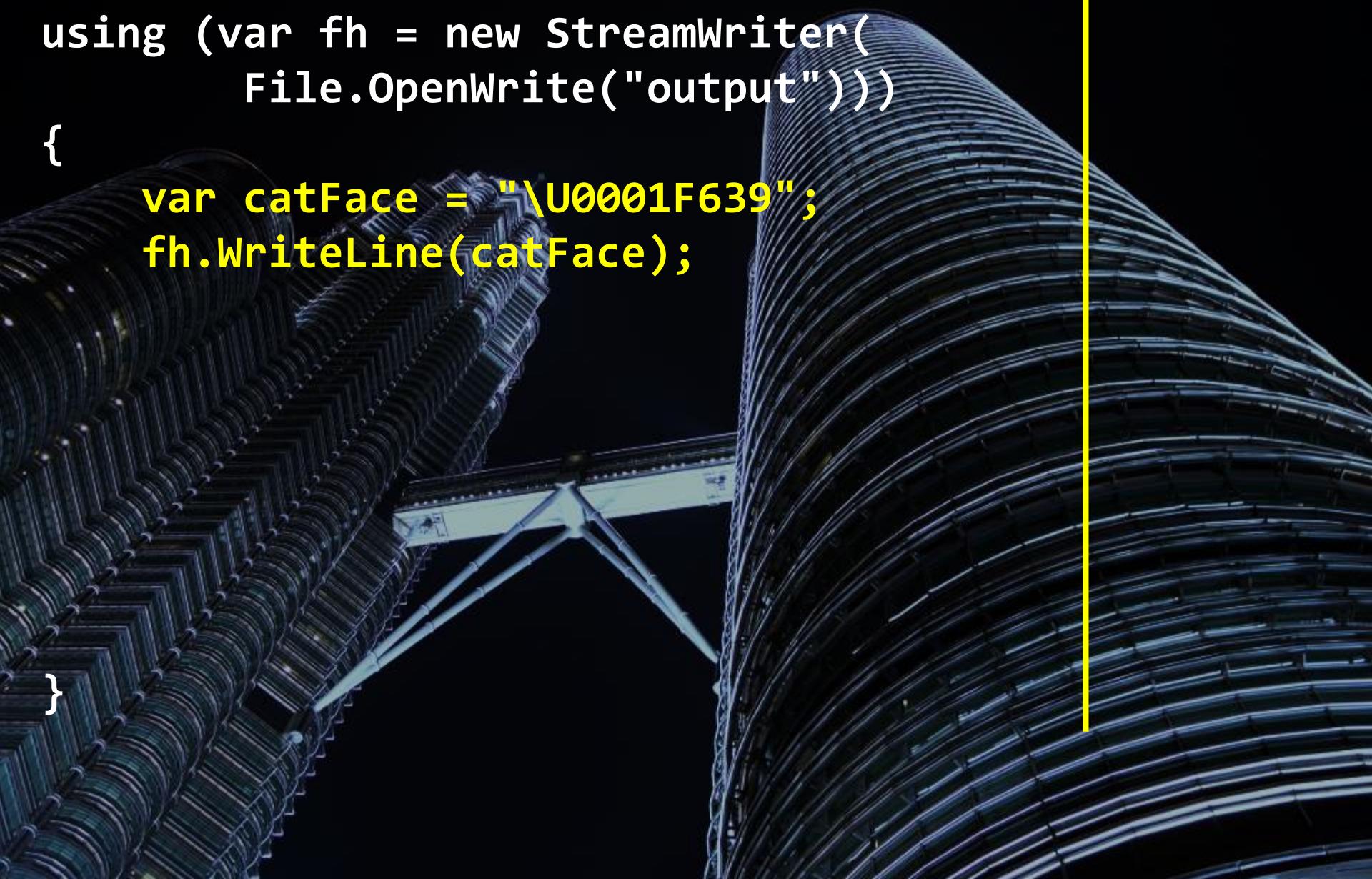
Đ

LATIN CAPITAL LETTER D  
COMBINING DOT BELOW  
COMBINING DOT ABOVE

The background image shows the Petronas Twin Towers at night, their iconic spires illuminated with a vibrant blue light. A walkway or bridge connects the two towers. The foreground is dark, making the bright yellow text stand out.

C#/Net

```
using (var fh = new StreamWriter(  
    File.OpenWrite("output")))  
{  
    var catFace = "\U0001F639";  
    fh.WriteLine(catFace);  
}  
}
```



```
using (var fh = new StreamWriter(  
    File.OpenWrite("output")))  
{  
    var catFace = "\U0001F639";  
    fh.WriteLine(catFace);  
}  
}
```



```
using (var fh = new StreamWriter(  
    File.OpenWrite("output")))  
{  
    var catFace = "\U0001F639";  
    fh.WriteLine(catFace);  
    fh.WriteLine(catFace.Length);  
}  
}
```



```
using (var fh = new StreamWriter(  
    File.OpenWrite("output")))  
{  
    var catFace = "\U0001F639";  
    fh.WriteLine(catFace);  
    fh.WriteLine(catFace.Length);  
}  
}
```



2

```
using (var fh = new StreamWriter(  
    File.OpenWrite("output")))  
{  
    var catFace = "\U0001F639";  
    fh.WriteLine(catFace);  
    fh.WriteLine(catFace.Length);  
  
    var dWithDots = "D\u0323\u0307";  
    fh.WriteLine(dWithDots);  
}
```



2

```
using (var fh = new StreamWriter(  
    File.OpenWrite("output")))  
{  
    var catFace = "\U0001F639";  
    fh.WriteLine(catFace);  
    fh.WriteLine(catFace.Length);  
  
    var dWithDots = "D\u0323\u0307";  
    fh.WriteLine(dWithDots);  
}
```



2  
đ

```
using (var fh = new StreamWriter(  
    File.OpenWrite("output")))  
{  
    var catFace = "\U0001F639";  
    fh.WriteLine(catFace);  
    fh.WriteLine(catFace.Length);  
  
    var dWithDots = "D\u0323\u0307";  
    fh.WriteLine(dWithDots);  
    fh.WriteLine(dWithDots.Length);  
}  
}
```



2  
D

```
using (var fh = new StreamWriter(  
    File.OpenWrite("output")))  
{  
    var catFace = "\U0001F639";  
    fh.WriteLine(catFace);  
    fh.WriteLine(catFace.Length);  
  
    var dWithDots = "D\u0323\u0307";  
    fh.WriteLine(dWithDots);  
    fh.WriteLine(dWithDots.Length);  
}  
}
```



2

D

3

```
using (var fh = new StreamWriter(  
    File.OpenWrite("output")))  
{  
    var catFace = "\U0001F639";  
    fh.WriteLine(catFace);  
    fh.WriteLine(catFace.Length);  
  
    var dWithDots = "D\u0323\u0307";  
    fh.WriteLine(dWithDots);  
    fh.WriteLine(dWithDots.Length);  
    dWithDots = dWithDots.Normalize();  
    fh.WriteLine(dWithDots.Length);  
}
```

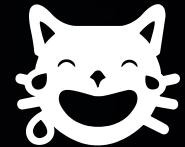


2

D

3

```
using (var fh = new StreamWriter(  
    File.OpenWrite("output")))  
{  
    var catFace = "\U0001F639";  
    fh.WriteLine(catFace);  
    fh.WriteLine(catFace.Length);  
  
    var dWithDots = "D\u0323\u0307";  
    fh.WriteLine(dWithDots);  
    fh.WriteLine(dWithDots.Length);  
    dWithDots = dWithDots.Normalize();  
    fh.WriteLine(dWithDots.Length);  
}
```



2

D.

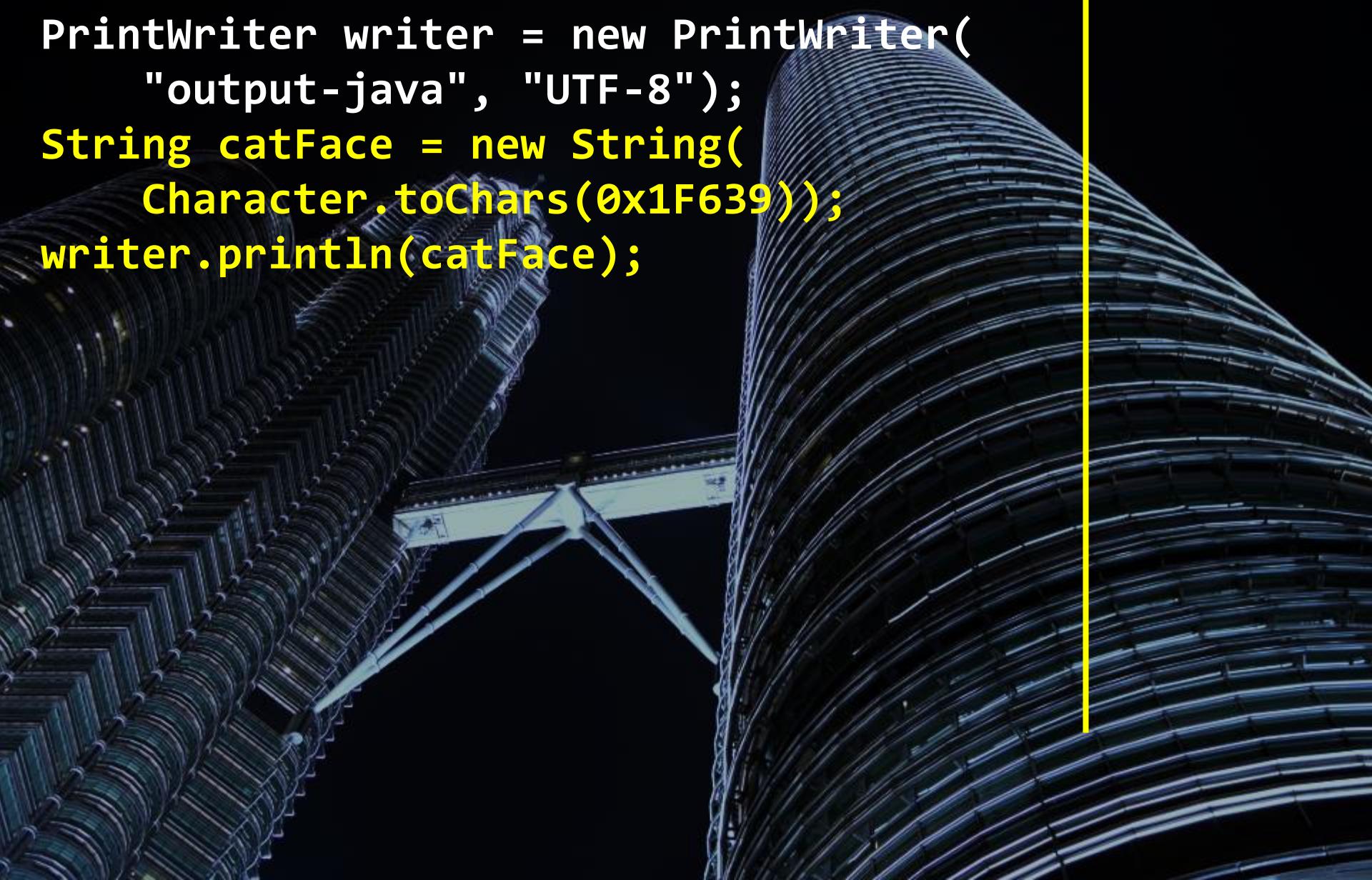
3

2

A night photograph of the Petronas Twin Towers in Kuala Lumpur, Malaysia. The towers are illuminated from within, creating a glowing effect against the dark night sky. The camera angle is looking up at the towers, emphasizing their height and unique curved, petal-like architecture.

**Java**

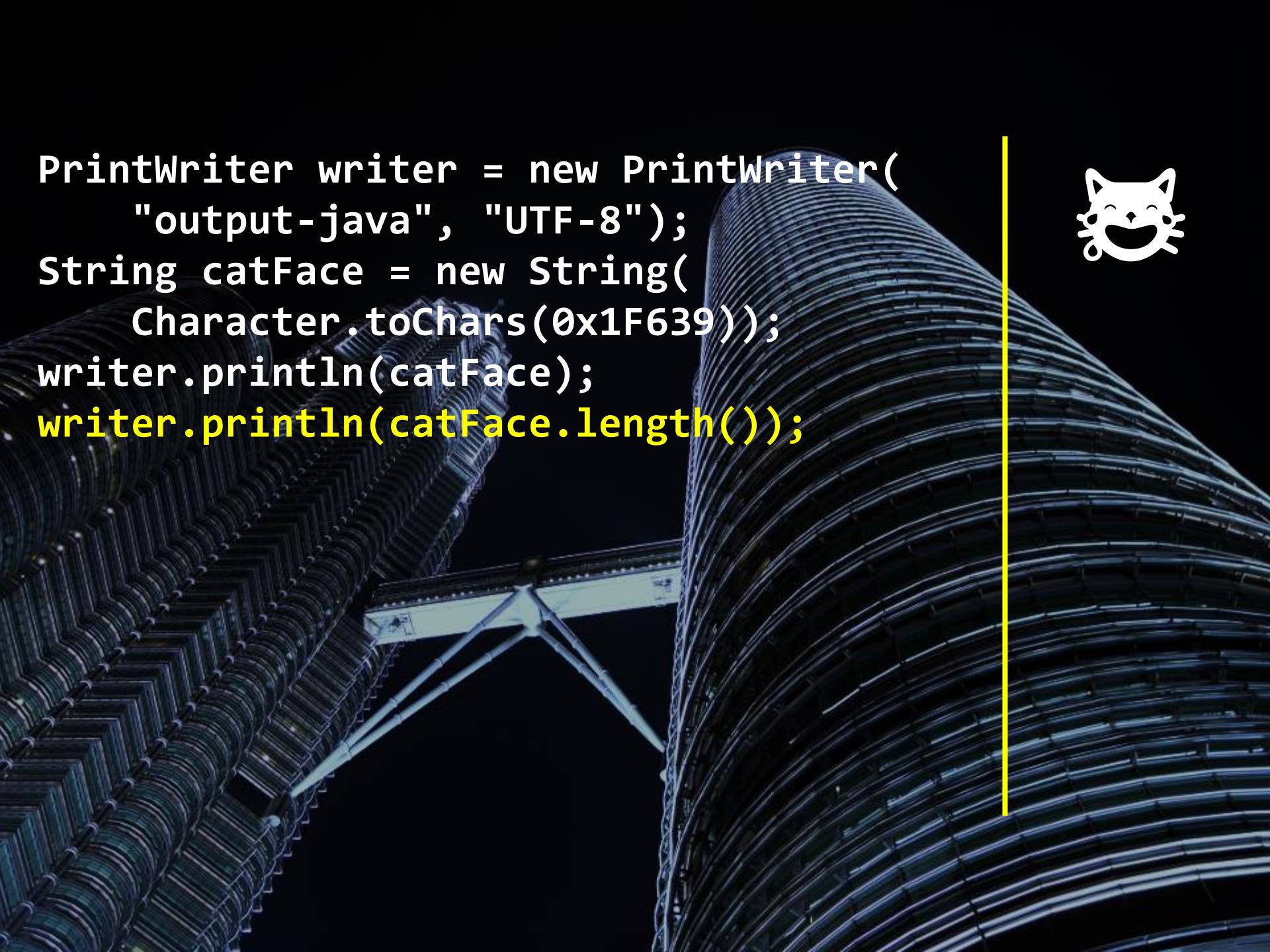
```
PrintWriter writer = new PrintWriter(  
    "output-java", "UTF-8");  
String catFace = new String(  
    Character.toChars(0x1F639));  
writer.println(catFace);
```



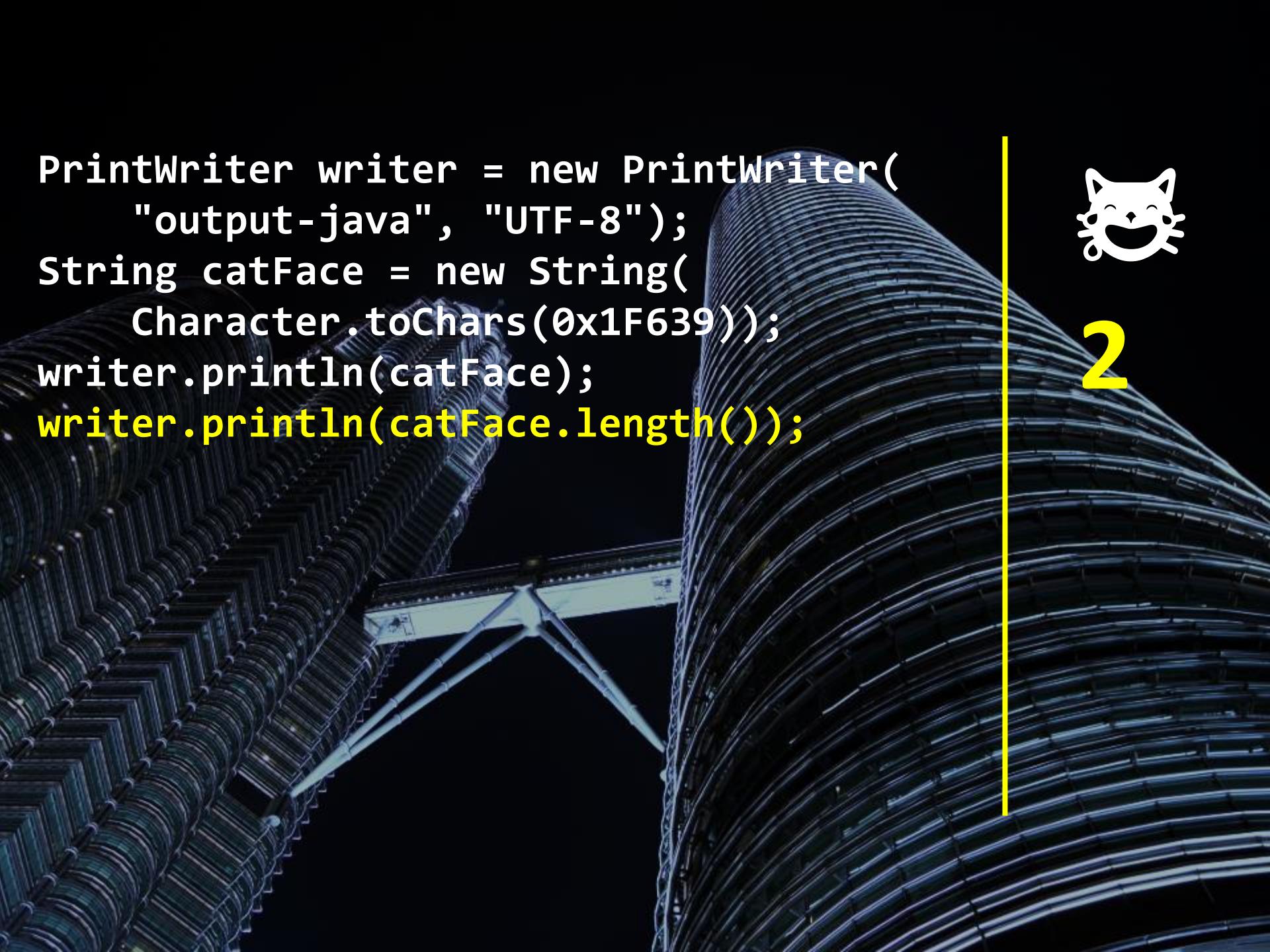
```
PrintWriter writer = new PrintWriter(  
    "output-java", "UTF-8");  
String catFace = new String(  
    Character.toChars(0x1F639));  
writer.println(catFace);
```



```
PrintWriter writer = new PrintWriter(  
    "output-java", "UTF-8");  
String catFace = new String(  
    Character.toChars(0x1F639));  
writer.println(catFace);  
writer.println(catFace.length());
```

A photograph of a modern architectural structure, likely the National Stadium in Beijing, known as the Bird's Nest. The building's exterior is composed of a complex, interlocking lattice of steel beams and columns, illuminated from within by a warm, golden light against a dark night sky. The perspective is from a low angle, looking up at the building's unique, organic shape.

```
PrintWriter writer = new PrintWriter(  
    "output-java", "UTF-8");  
String catFace = new String(  
    Character.toChars(0x1F639));  
writer.println(catFace);  
writer.println(catFace.length());
```

A photograph of a modern architectural structure, likely the National Stadium in Beijing, known as the Bird's Nest. The building's distinctive curved, ribbed facade is illuminated from within, creating a glowing effect against the dark night sky. The central opening of the stadium is also visible.

2

```
PrintWriter writer = new PrintWriter(  
    "output-java", "UTF-8");  
String catFace = new String(  
    Character.toChars(0x1F639));  
writer.println(catFace);  
writer.println(catFace.length());
```

```
String dWithDots = "D\u0323\u0307";  
writer.println(dWithDots);
```



2

```
PrintWriter writer = new PrintWriter(  
    "output-java", "UTF-8");  
String catFace = new String(  
    Character.toChars(0x1F639));  
writer.println(catFace);  
writer.println(catFace.length());
```

```
String dWithDots = "D\u0323\u0307";  
writer.println(dWithDots);
```



2  
đ

```
PrintWriter writer = new PrintWriter(  
    "output-java", "UTF-8");  
String catFace = new String(  
    Character.toChars(0x1F639));  
writer.println(catFace);  
writer.println(catFace.length());  
  
String dWithDots = "D\u0323\u0307";  
writer.println(dWithDots);  
writer.println(dWithDots.length());
```



2  
D

```
PrintWriter writer = new PrintWriter(  
    "output-java", "UTF-8");  
String catFace = new String(  
    Character.toChars(0x1F639));  
writer.println(catFace);  
writer.println(catFace.length());  
  
String dWithDots = "D\u0323\u0307";  
writer.println(dWithDots);  
writer.println(dWithDots.length());
```



2

D

3

```
PrintWriter writer = new PrintWriter(  
    "output-java", "UTF-8");  
String catFace = new String(  
    Character.toChars(0x1F639));  
writer.println(catFace);  
writer.println(catFace.length());  
  
String dWithDots = "D\u0323\u0307";  
writer.println(dWithDots);  
writer.println(dWithDots.length());  
dWithDots = Normalizer.normalize(  
    dWithDots, Normalizer.Form.NFC);  
writer.println(dWithDots.length());
```



2

D.

3

```
PrintWriter writer = new PrintWriter(  
    "output-java", "UTF-8");  
String catFace = new String(  
    Character.toChars(0x1F639));  
writer.println(catFace);  
writer.println(catFace.length());  
  
String dWithDots = "D\u0323\u0307";  
writer.println(dWithDots);  
writer.println(dWithDots.length());  
dWithDots = Normalizer.normalize(  
    dWithDots, Normalizer.Form.NFC);  
writer.println(dWithDots.length());
```



2

D.

3

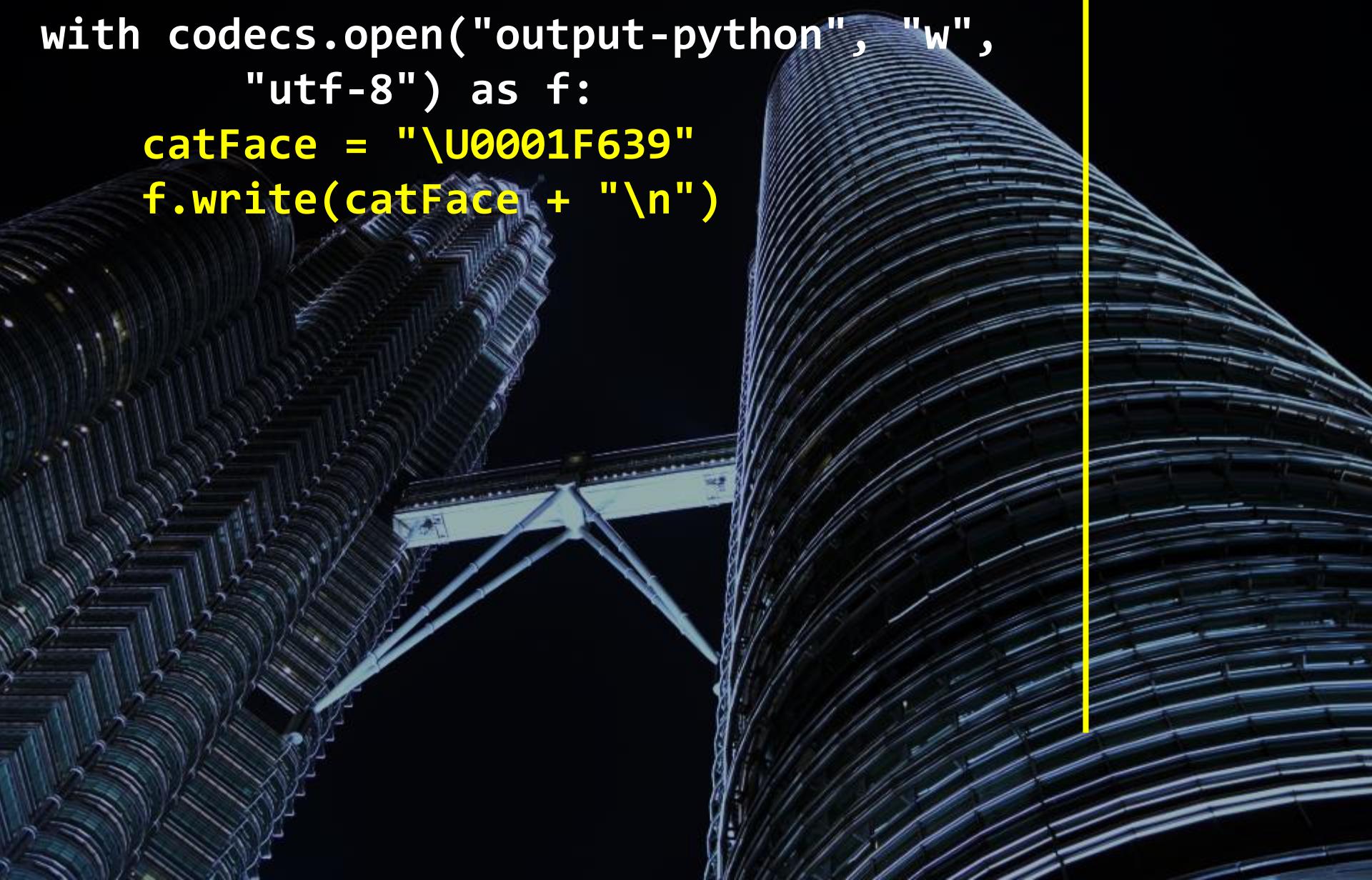
2



# Python

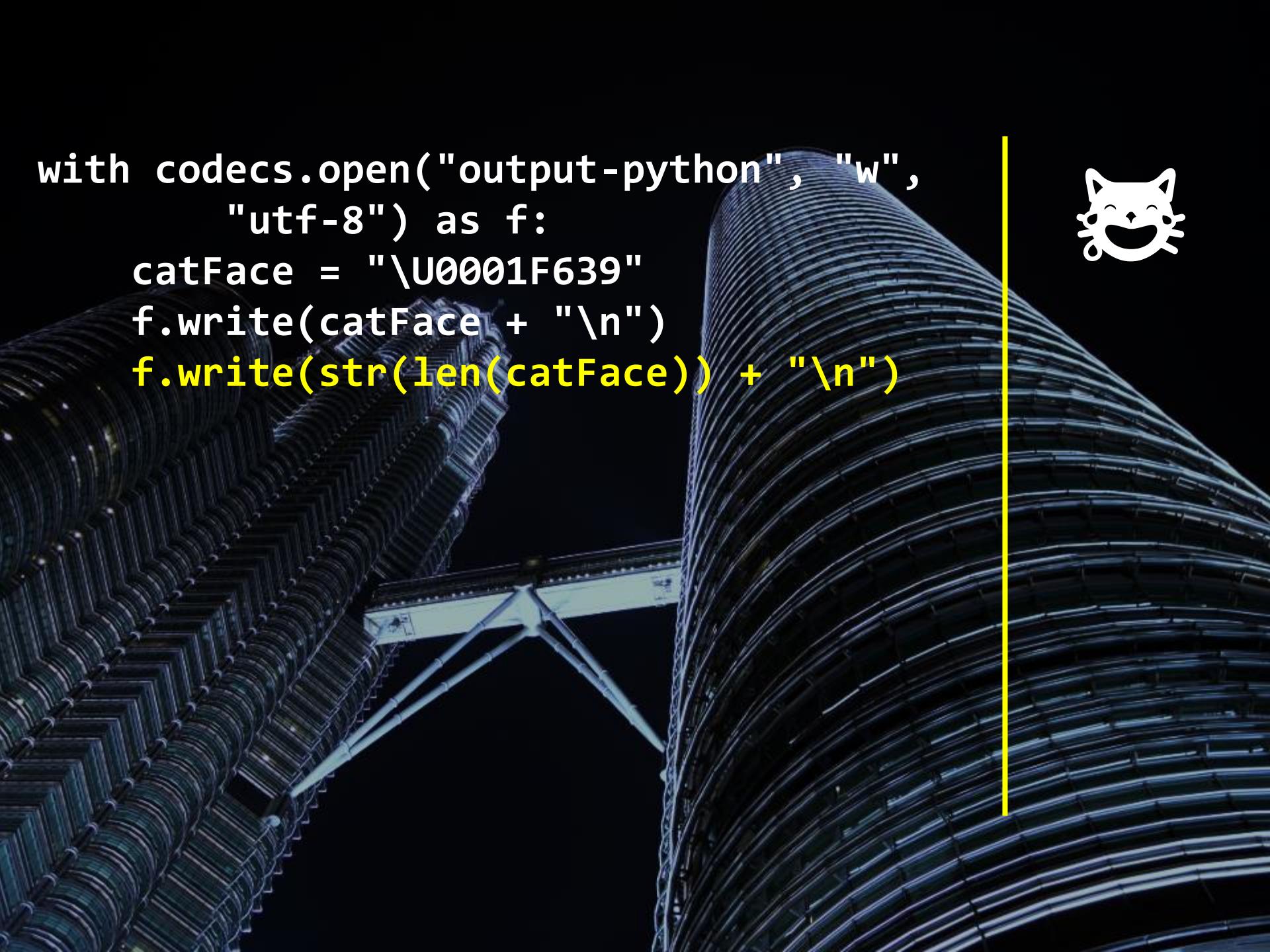
(3, with gooder Unicode)

```
with codecs.open("output-python", "w",  
    "utf-8") as f:  
    catFace = "\U0001F639"  
    f.write(catFace + "\n")
```



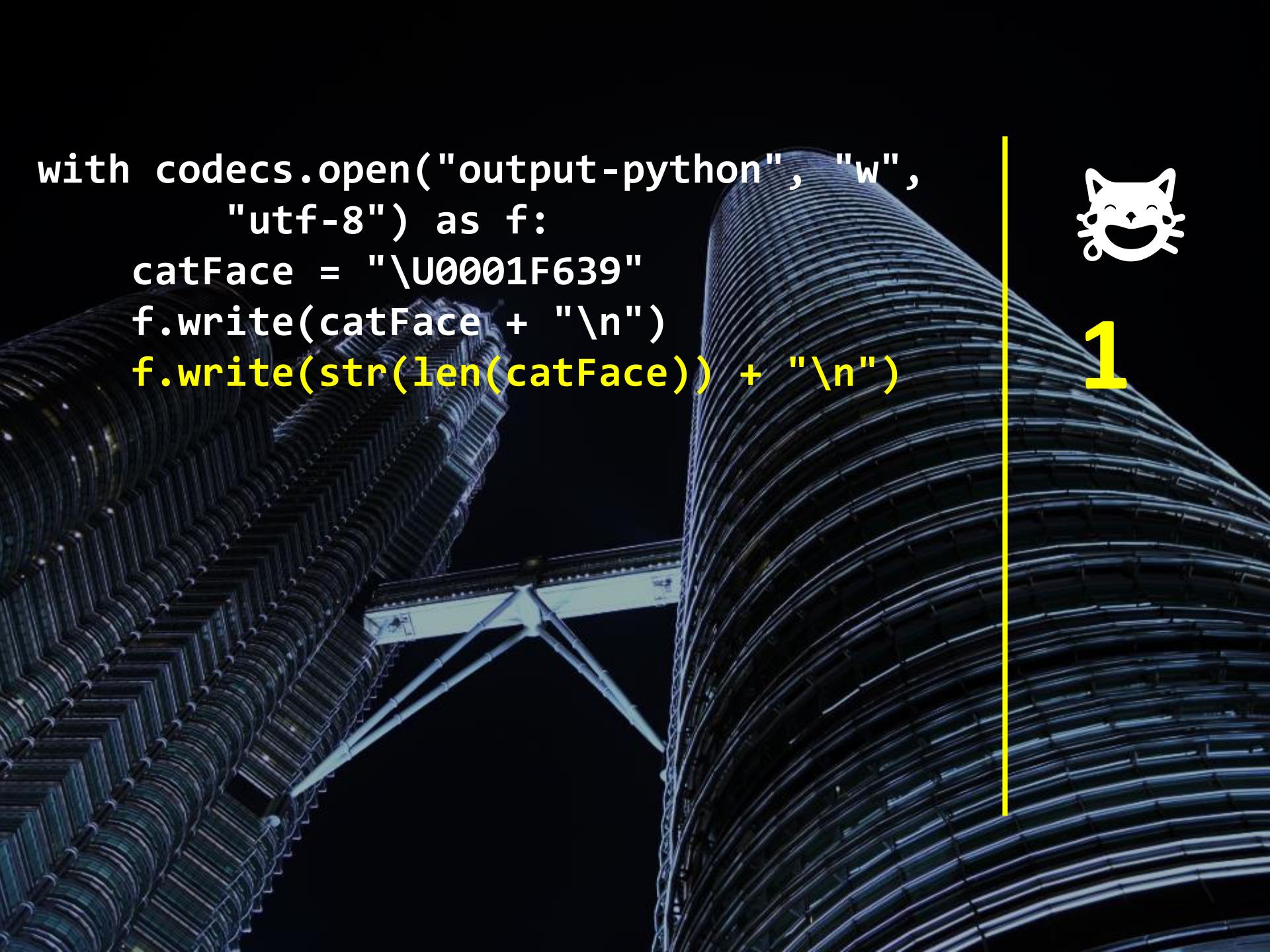
```
with codecs.open("output-python", "w",  
    "utf-8") as f:  
    catFace = "\U0001F639"  
    f.write(catFace + "\n")
```





```
with codecs.open("output-python", "w",
    "utf-8") as f:
    catFace = "\U0001F639"
    f.write(catFace + "\n")
    f.write(str(len(catFace)) + "\n")
```

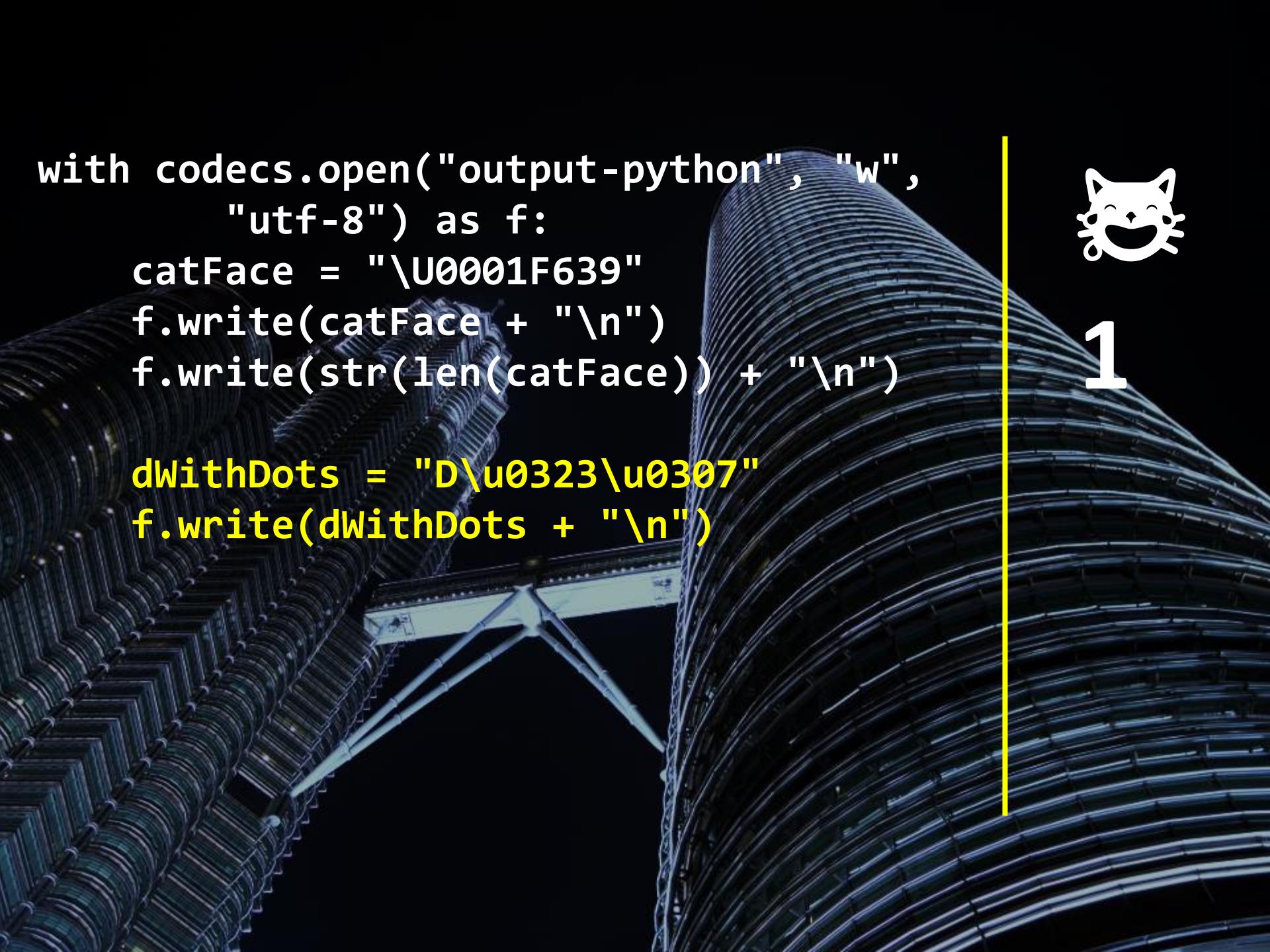




```
with codecs.open("output-python", "w",
    "utf-8") as f:
    catFace = "\U0001F639"
    f.write(catFace + "\n")
    f.write(str(len(catFace)) + "\n")
```



1



```
with codecs.open("output-python", "w",
                 "utf-8") as f:
    catFace = "\U0001F639"
    f.write(catFace + "\n")
    f.write(str(len(catFace)) + "\n")
```

```
dWithDots = "D\u0323\u0307"
f.write(dWithDots + "\n")
```



1



```
with codecs.open("output-python", "w",
                 "utf-8") as f:
    catFace = "\U0001F639"
    f.write(catFace + "\n")
    f.write(str(len(catFace)) + "\n")

dWithDots = "D\u0323\u0307"
f.write(dWithDots + "\n")
```



1  
đ

```
with codecs.open("output-python", "w",
                 "utf-8") as f:
    catFace = "\U0001F639"
    f.write(catFace + "\n")
    f.write(str(len(catFace)) + "\n")

    dWithDots = "D\u0323\u0307"
    f.write(dWithDots + "\n")
    f.write(str(len(dWithDots)) + "\n")
```



1

Đ

```
with codecs.open("output-python", "w",
                 "utf-8") as f:
    catFace = "\U0001F639"
    f.write(catFace + "\n")
    f.write(str(len(catFace)) + "\n")

    dWithDots = "D\u0323\u0307"
    f.write(dWithDots + "\n")
    f.write(str(len(dWithDots)) + "\n")
```



1  
D  
3

```
with codecs.open("output-python", "w",
                 "utf-8") as f:
    catFace = "\U0001F639"
    f.write(catFace + "\n")
    f.write(str(len(catFace)) + "\n")

    dWithDots = "D\u0323\u0307"
    f.write(dWithDots + "\n")
    f.write(str(len(dWithDots)) + "\n")
    dWithDots = unicodedata.normalize(
        'NFC', dWithDots)
    f.write(str(len(dWithDots)) + "\n")
```



1  
D.  
3

```
with codecs.open("output-python", "w",
                 "utf-8") as f:
    catFace = "\U0001F639"
    f.write(catFace + "\n")
    f.write(str(len(catFace)) + "\n")

    dWithDots = "D\u0323\u0307"
    f.write(dWithDots + "\n")
    f.write(str(len(dWithDots)) + "\n")
    dWithDots = unicodedata.normalize(
        'NFC', dWithDots)
    f.write(str(len(dWithDots)) + "\n")
```

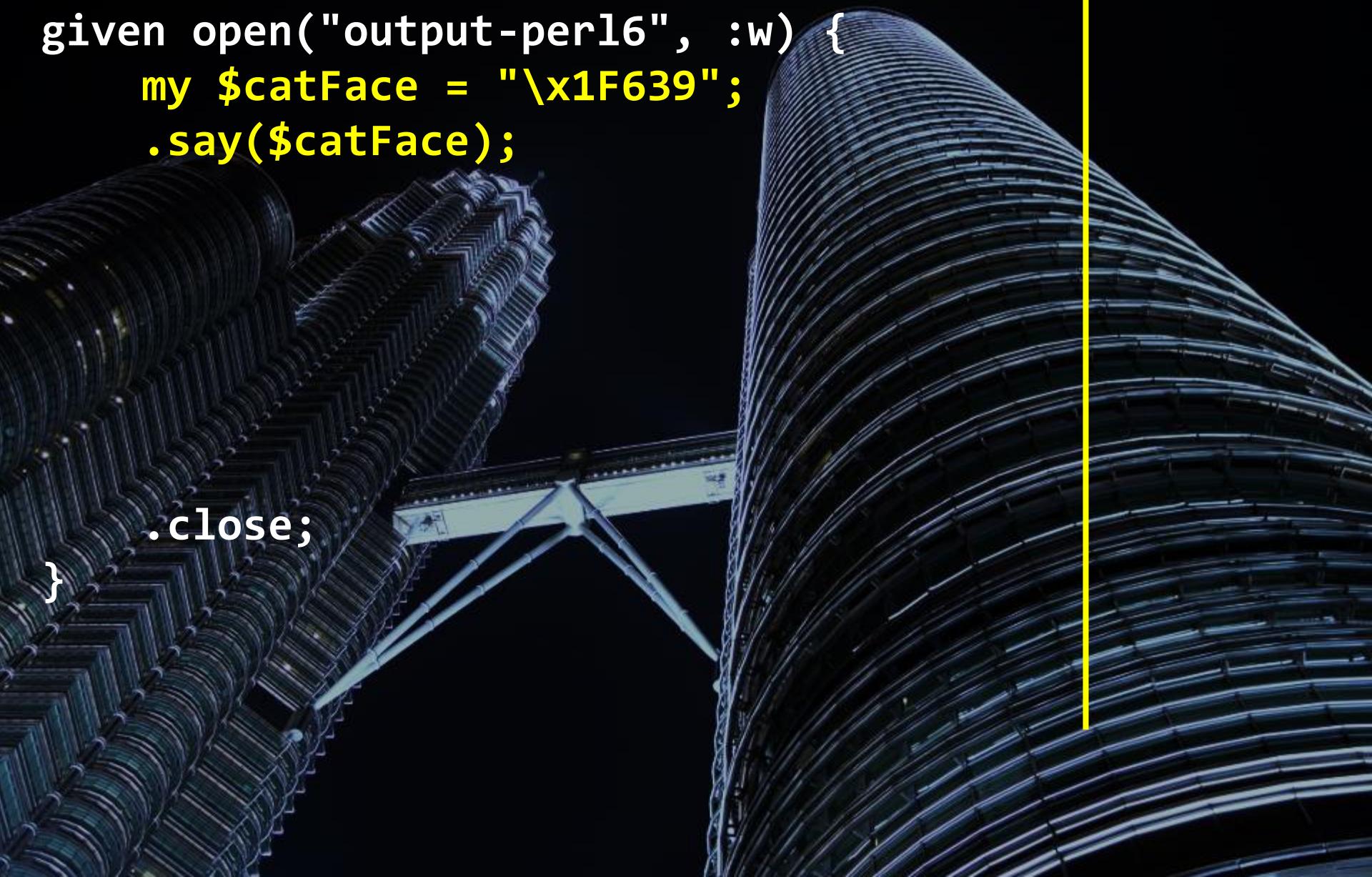


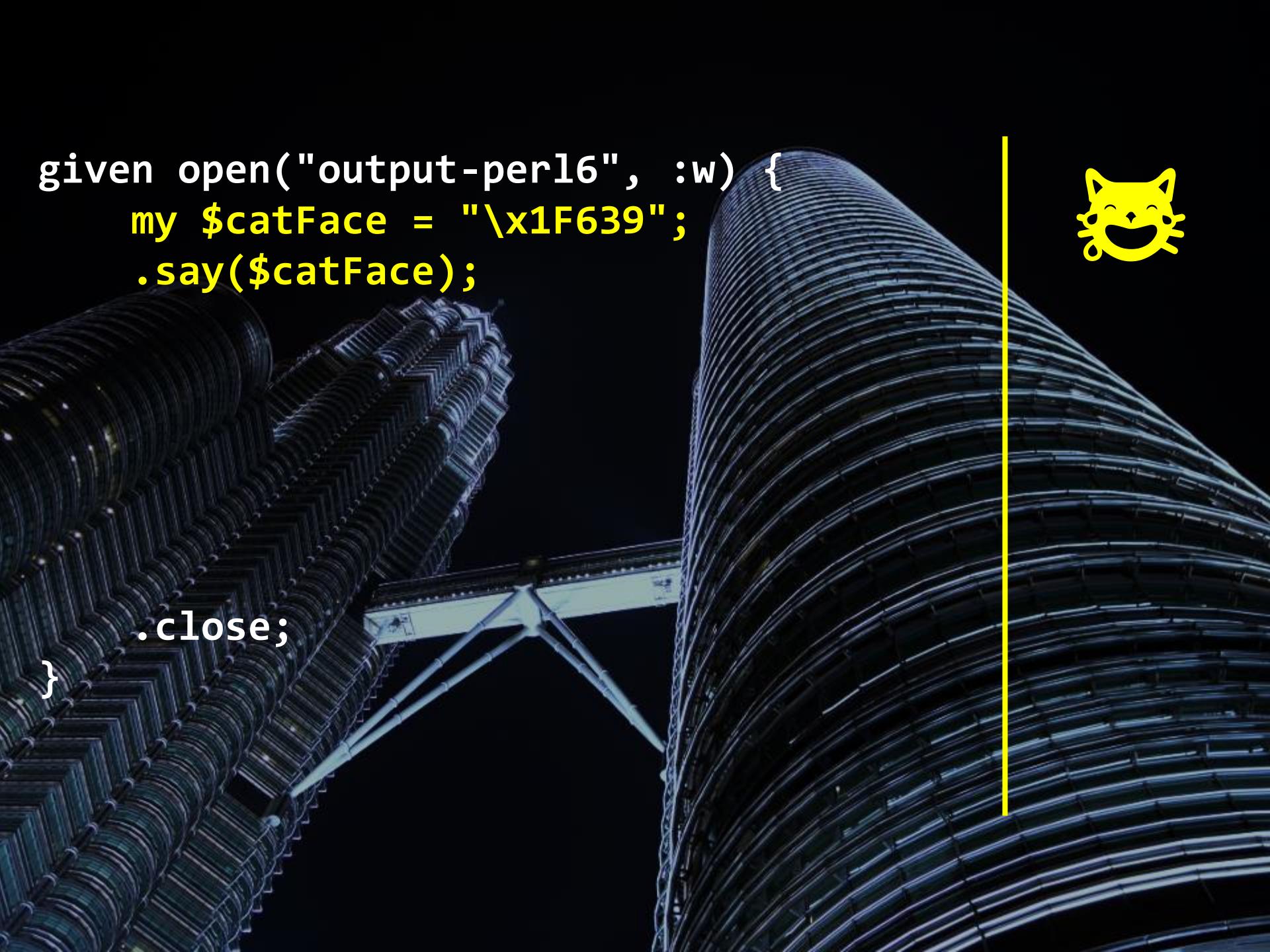
1  
Đ  
3  
2



Perl 6

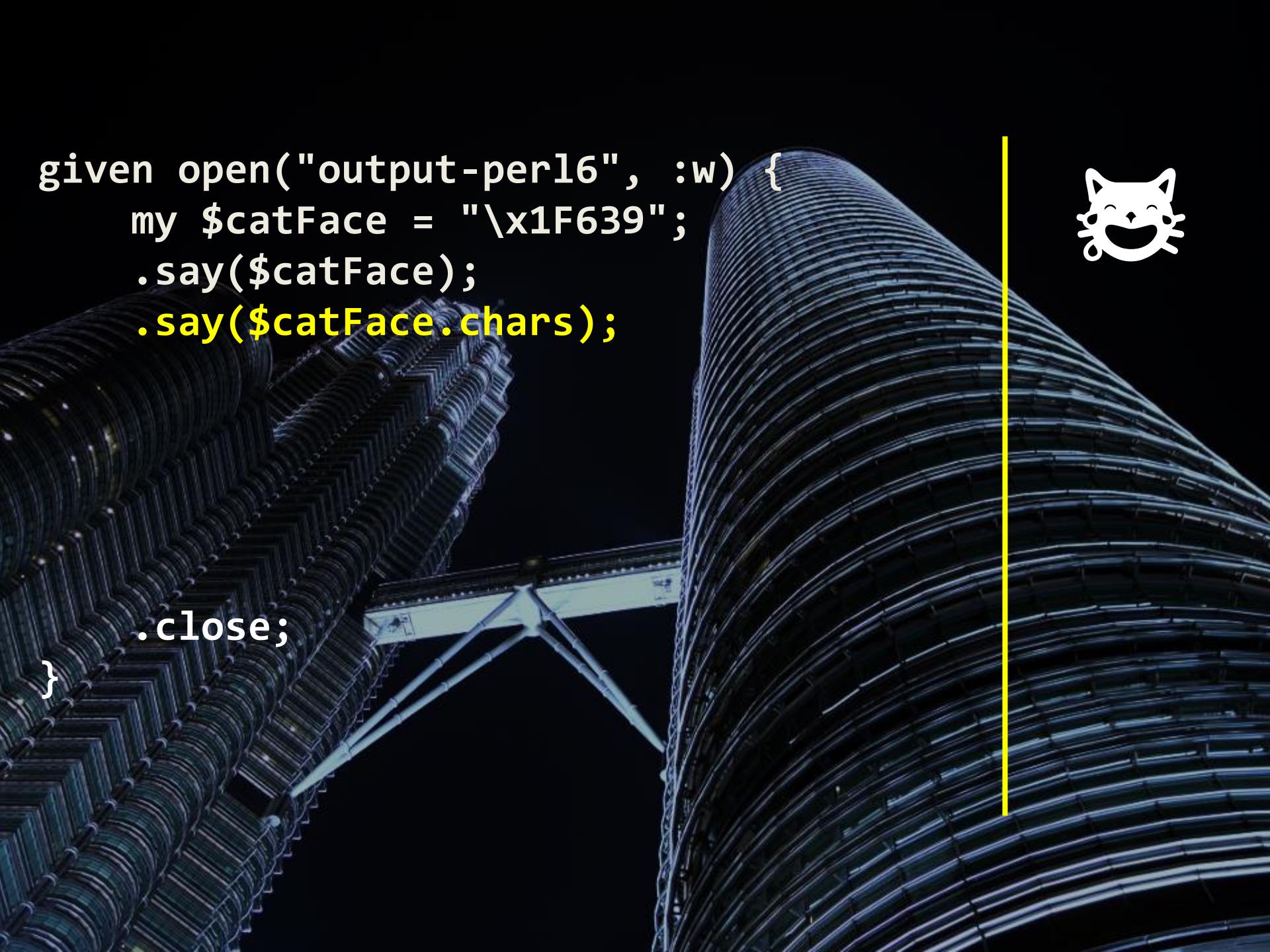
```
given open("output-perl6", :w) {  
    my $catFace = "\xF1F639";  
    .say($catFace);  
  
    .close;  
}
```



A photograph of a modern architectural structure, likely the Petronas Twin Towers, viewed from a low angle looking up. The building's distinctive curved, petal-like facade is illuminated with blue lights against a dark night sky. A yellow vertical line is drawn down the right side of the image.

```
given open("output-perl6", :w) {  
    my $catFace = "\xF1F639";  
    .say($catFace);  
  
    .close;  
}
```



A photograph of a modern architectural structure, likely the Petronas Twin Towers, taken from a low angle at night. The building's distinctive curved facade is illuminated with blue lights, creating a glowing effect against the dark sky. The central entrance area is brightly lit.

```
given open("output-perl6", :w) {  
    my $catFace = "\xF0\x9F\x98\x89";  
    .say($catFace);  
    .say($catFace.chars);  
  
    .close;  
}
```



A photograph of a modern architectural structure, likely the Petronas Twin Towers, viewed from a low angle looking up. The building's distinctive curved facade is illuminated with blue lights, creating a glowing effect against the dark night sky. The central entrance area is brightly lit.

```
given open("output-perl6", :w) {  
    my $catFace = "\xF0\x9F\x98\x89";  
    .say($catFace);  
    .say($catFace.chars);  
  
    .close;  
}
```



1

The background of the slide features a photograph of a modern architectural structure, likely the Burj Khalifa, with its distinctive curved facade and illuminated windows.

```
given open("output-perl6", :w) {
    my $catFace = "\xF1F639";
    .say($catFace);
    .say($catFace.chars);

    my $dWithDots = "D\x0323\x0307";
    .say($dWithDots);

    .close;
}
```



1



```
given open("output-perl6", :w) {
    my $catFace = "\xF1F639";
    .say($catFace);
    .say($catFace.chars);

    my $dWithDots = "D\x0323\x0307";
    .say($dWithDots);

    .close;
}
```



1  
đ

```
given open("output-perl6", :w) {  
    my $catFace = "\xF1F639";  
    .say($catFace);  
    .say($catFace.chars);  
  
    my $dWithDots = "D\x0323\x0307";  
    .say($dWithDots);  
    .say($dWithDots.chars);  
  
    .close;  
}
```



1  
D

```
given open("output-perl6", :w) {  
    my $catFace = "\xF1F639";  
    .say($catFace);  
    .say($catFace.chars);  
  
    my $dWithDots = "D\x0323\x0307";  
    .say($dWithDots);  
    .say($dWithDots.chars);  
  
    .close;  
}
```



1

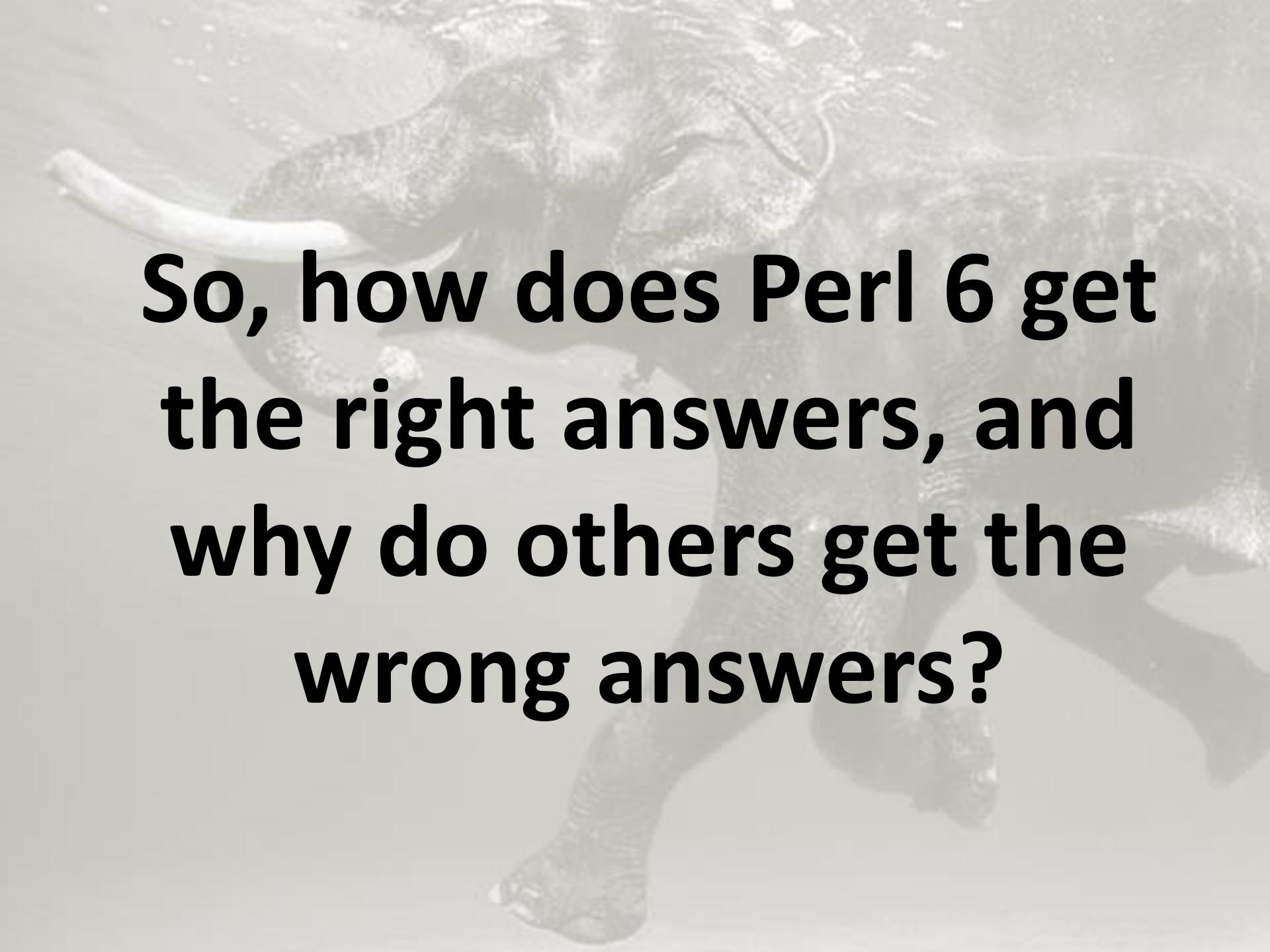
D.

1

U+270C

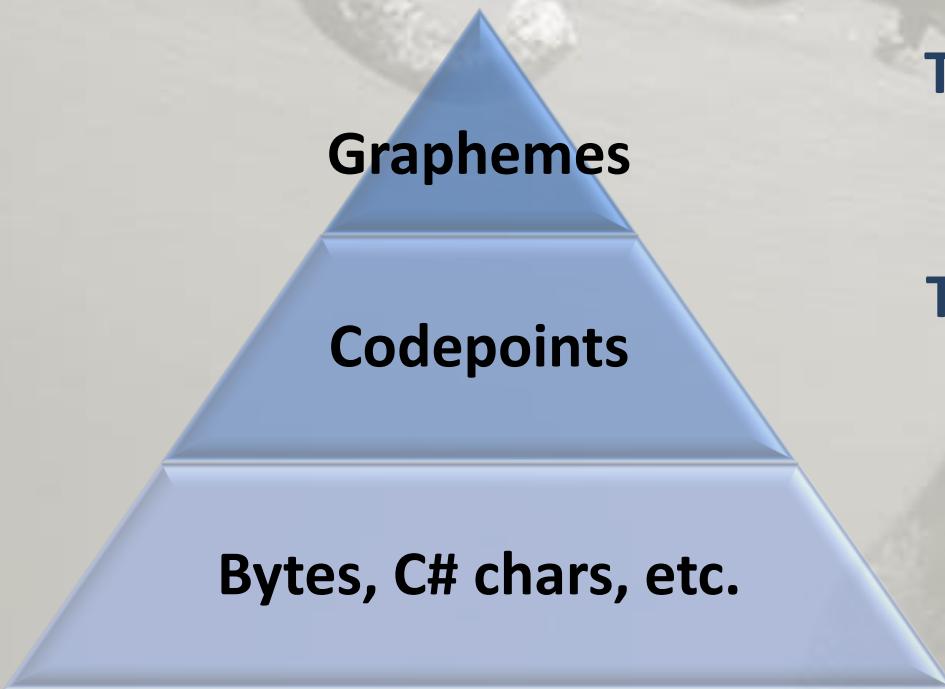


VICTORY HAND

A black and white photograph showing a person's lower body from the side. They are wearing dark trousers and sandals, walking across a sandy surface. The background is slightly blurred.

**So, how does Perl 6 get  
the right answers, and  
why do others get the  
wrong answers?**

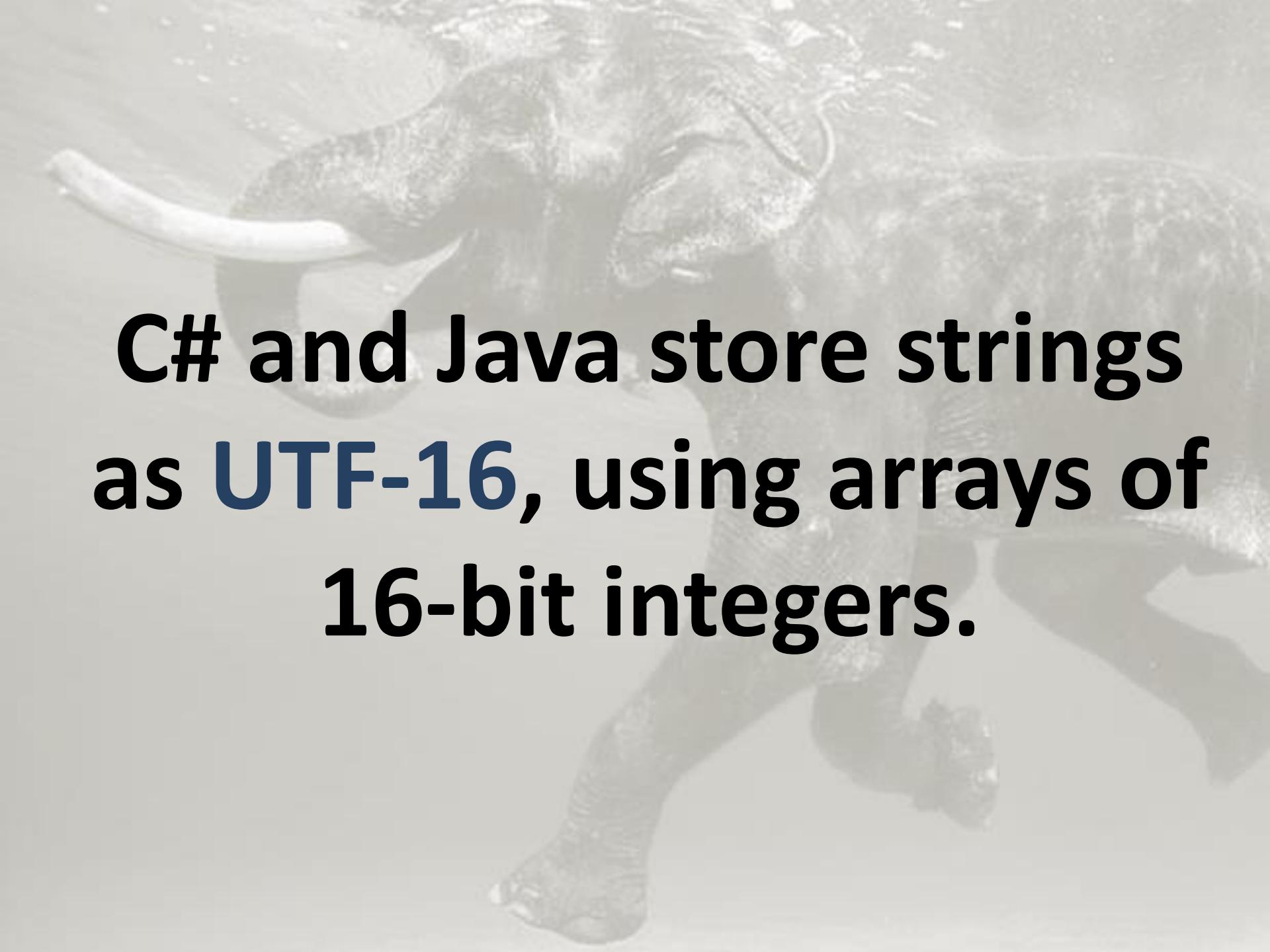
# The 3 levels of Unicode



Things a human would consider  
a character

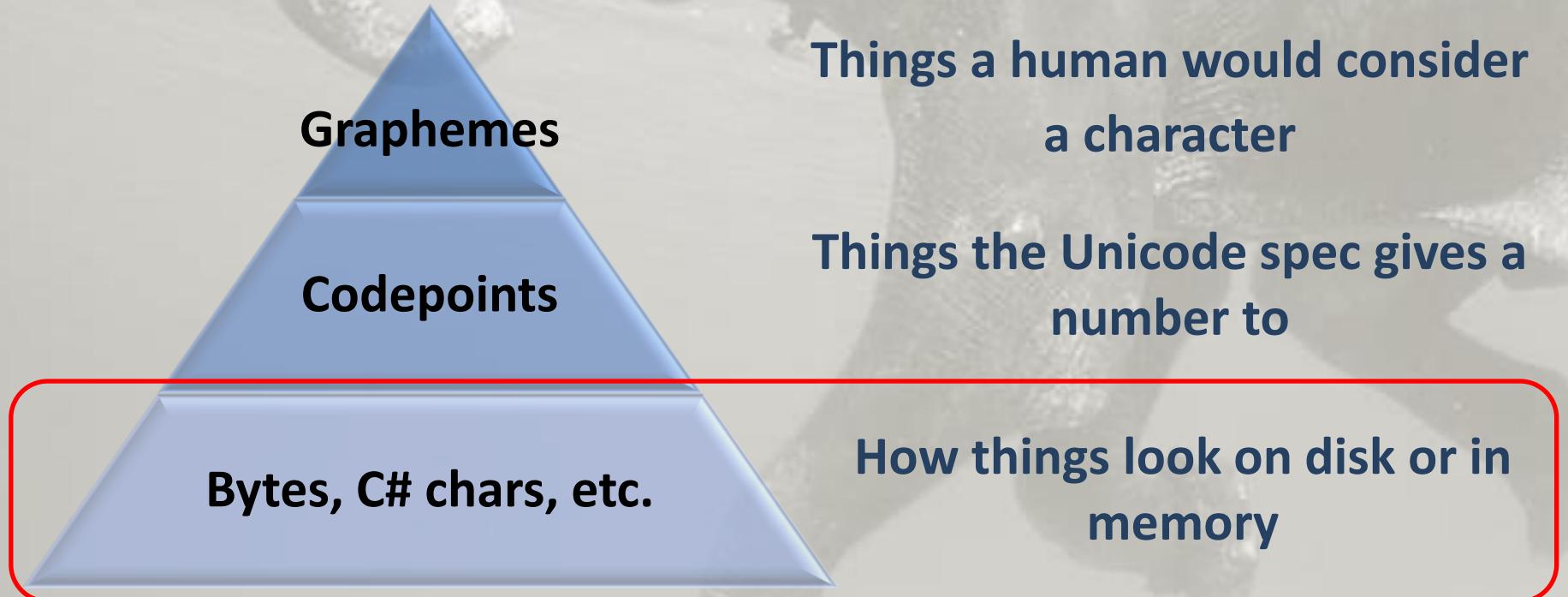
Things the Unicode spec gives a  
number to

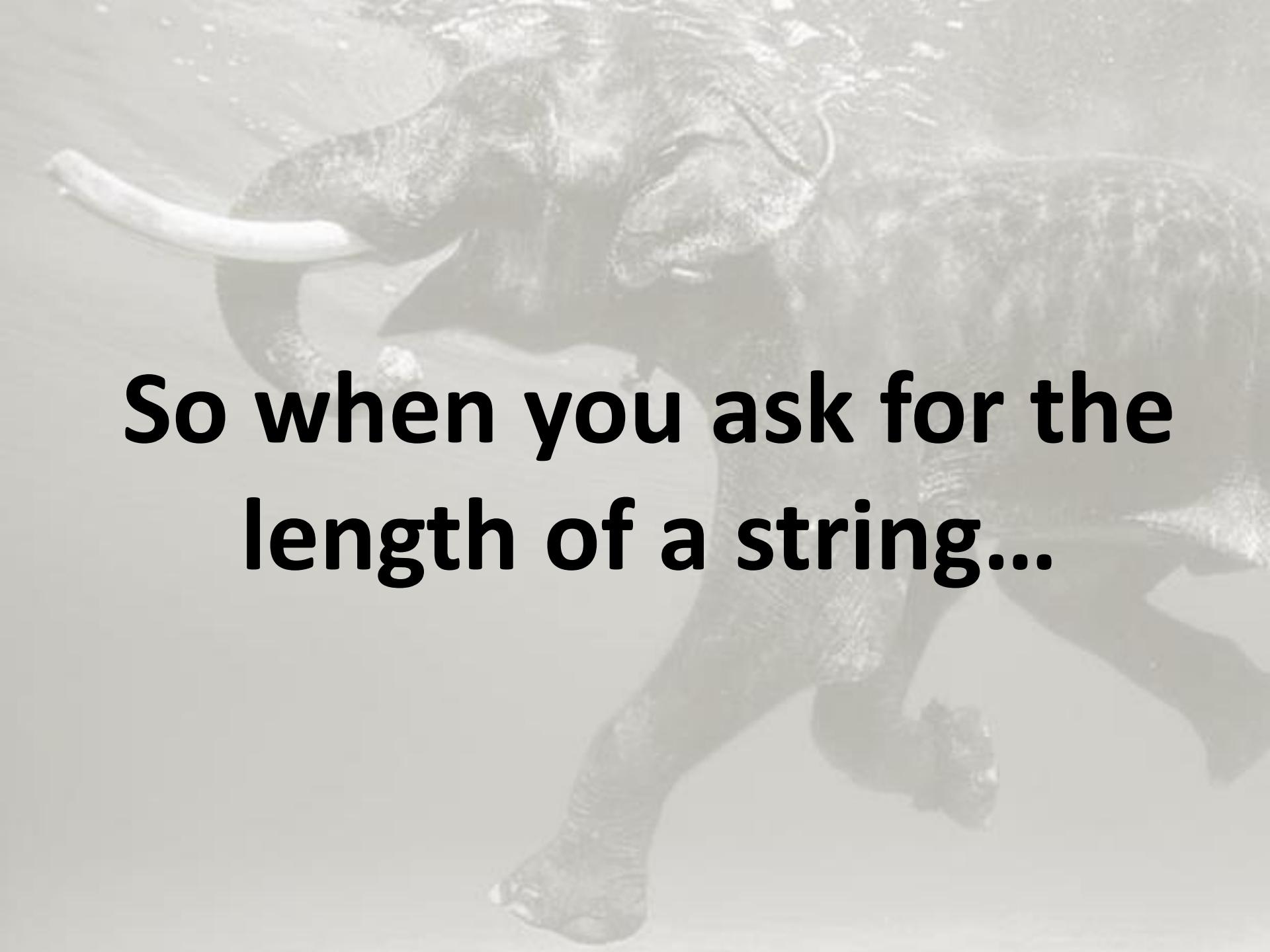
How things look on disk or in  
memory



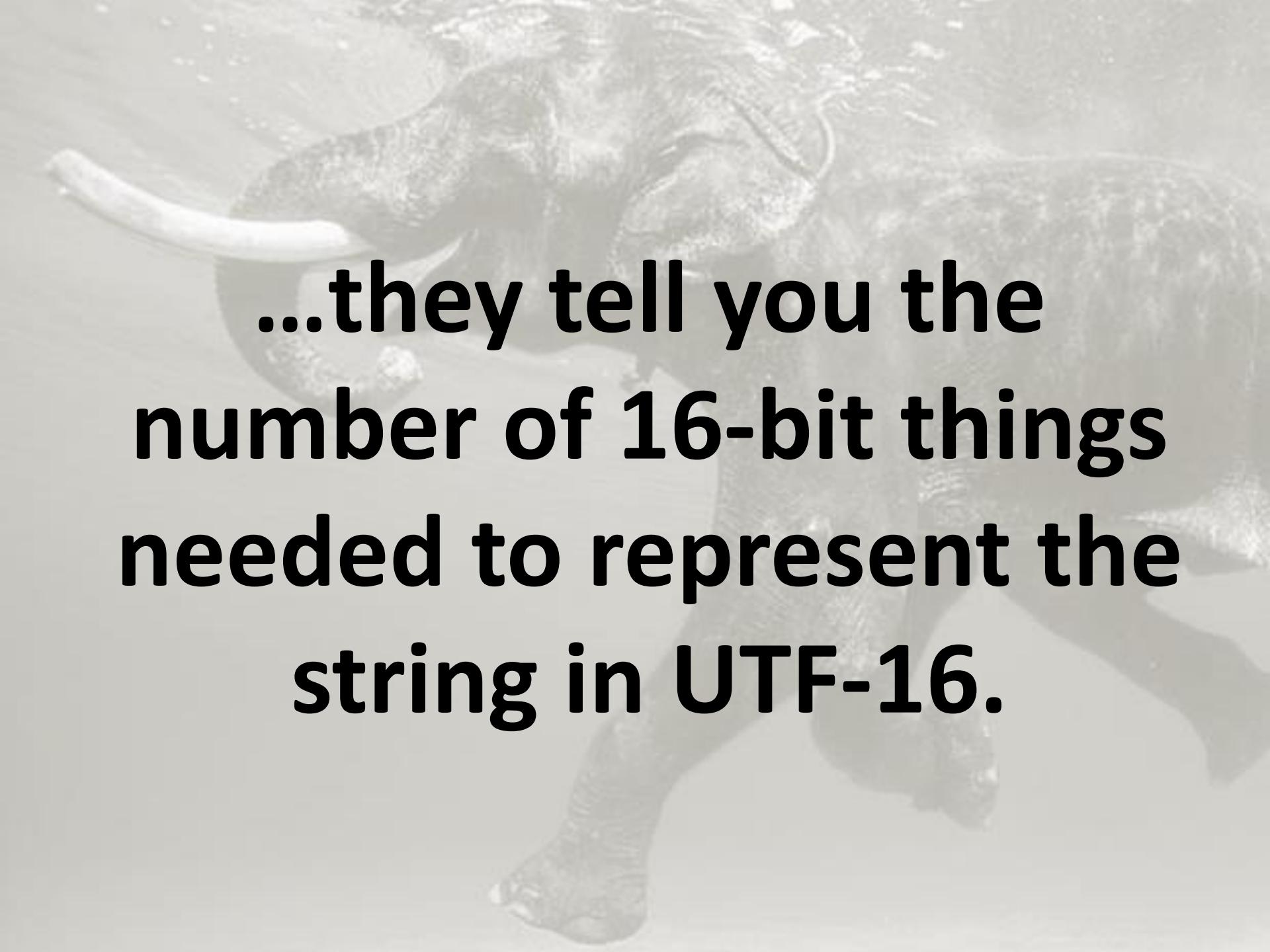
**C# and Java store strings  
as UTF-16, using arrays of  
16-bit integers.**

# They work down here!





**So when you ask for the  
length of a string...**



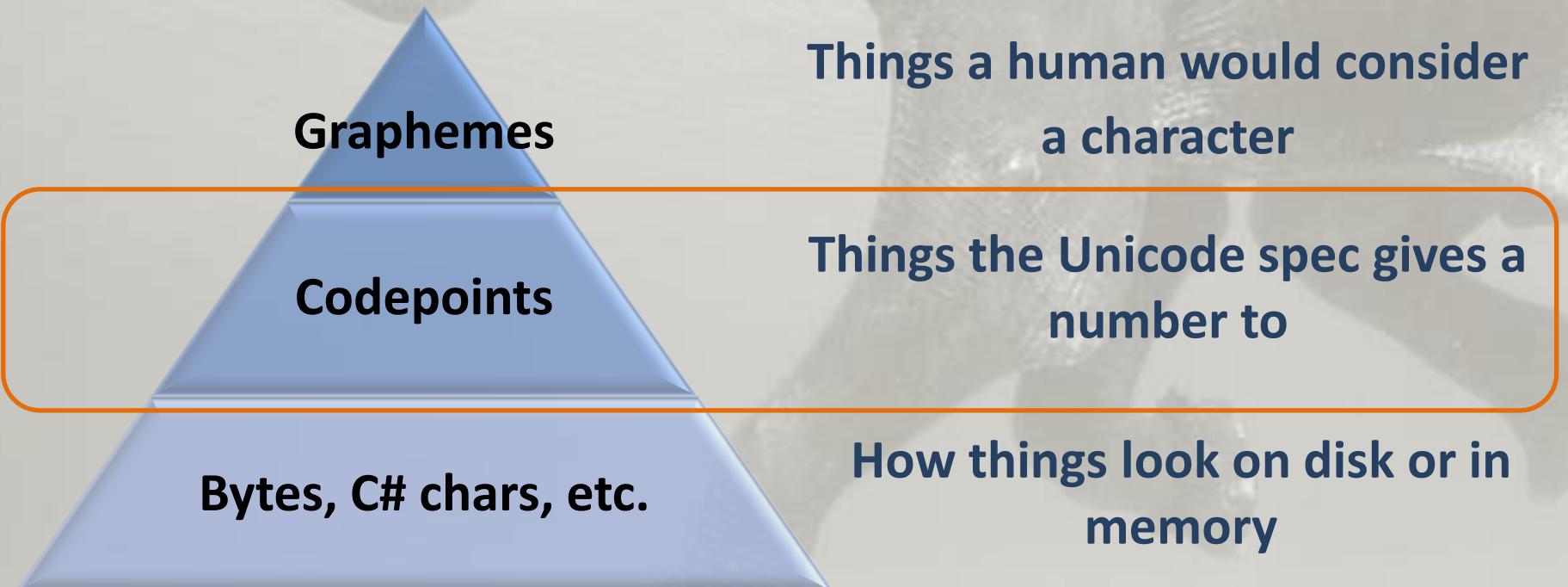
**...they tell you the  
number of 16-bit things  
needed to represent the  
string in UTF-16.**

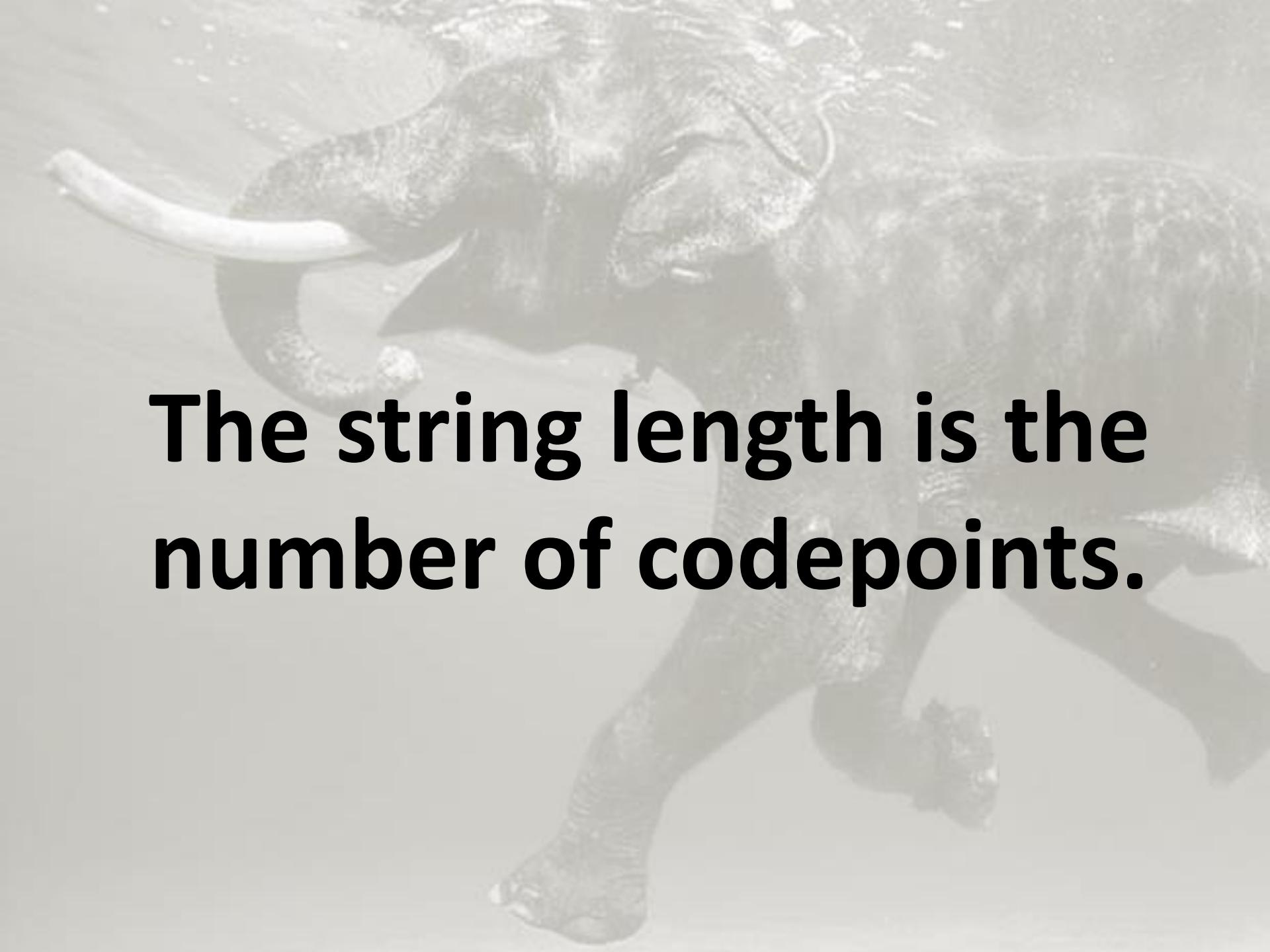


**It's...enterprise.**



# Some languages have codepoint-level strings



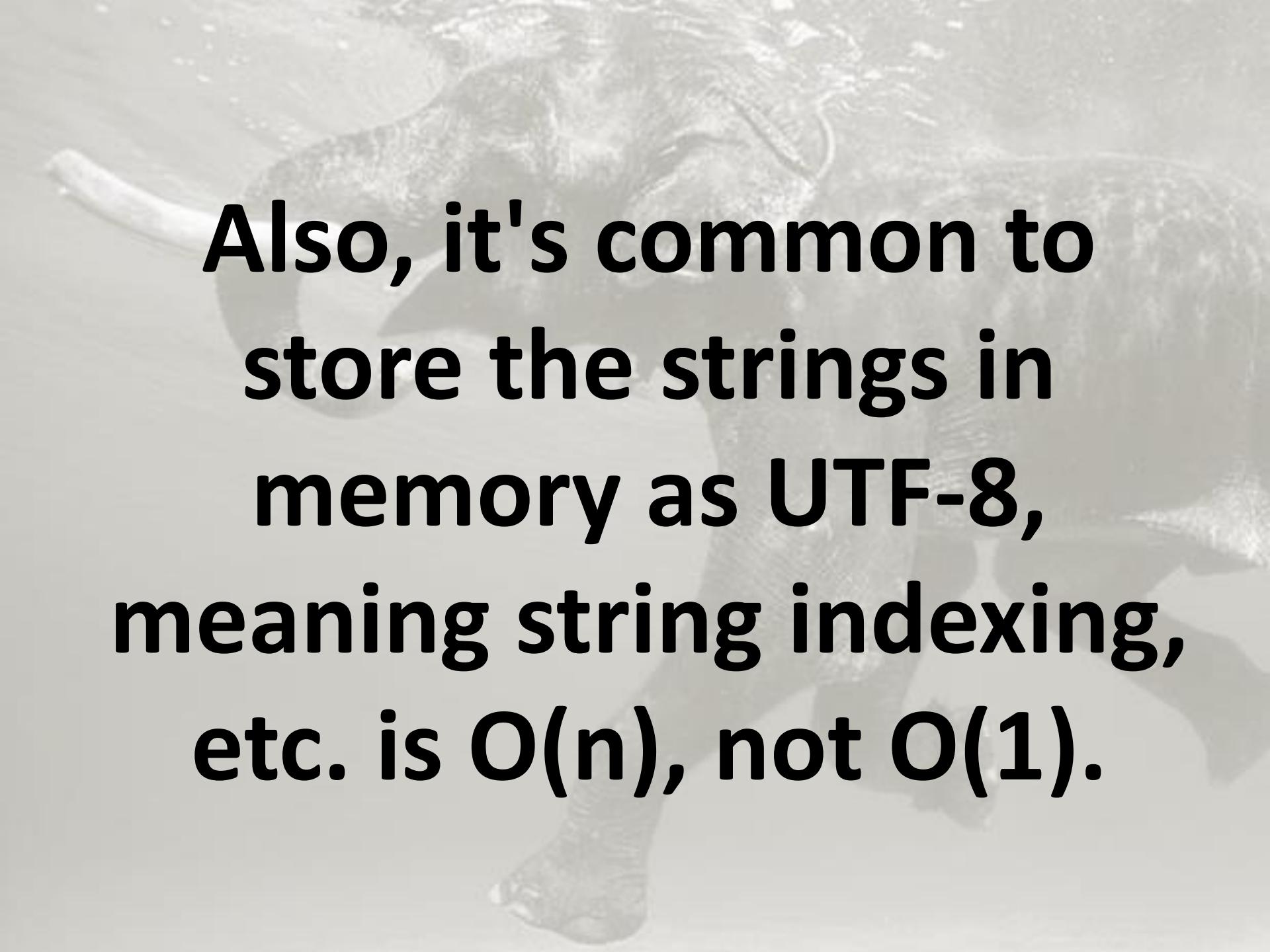


The string length is the number of codepoints.



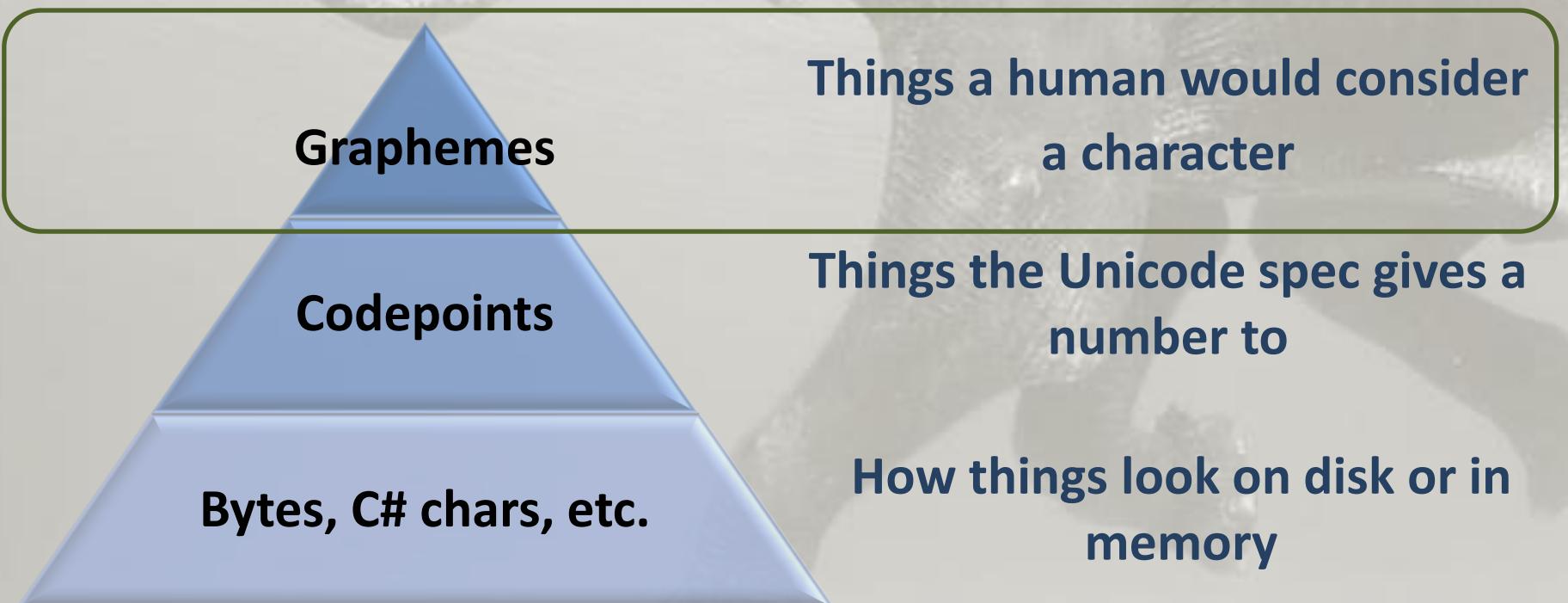
**Which is close, but...**





Also, it's common to store the strings in memory as UTF-8, meaning string indexing, etc. is  $O(n)$ , not  $O(1)$ .

# Perl 6 strings work at the grapheme level



Đ

LATIN CAPITAL LETTER D  
COMBINING DOT BELOW  
COMBINING DOT ABOVE

=

3 codepoints

=

2 normalized (NFC) codepoints

=

1 grapheme

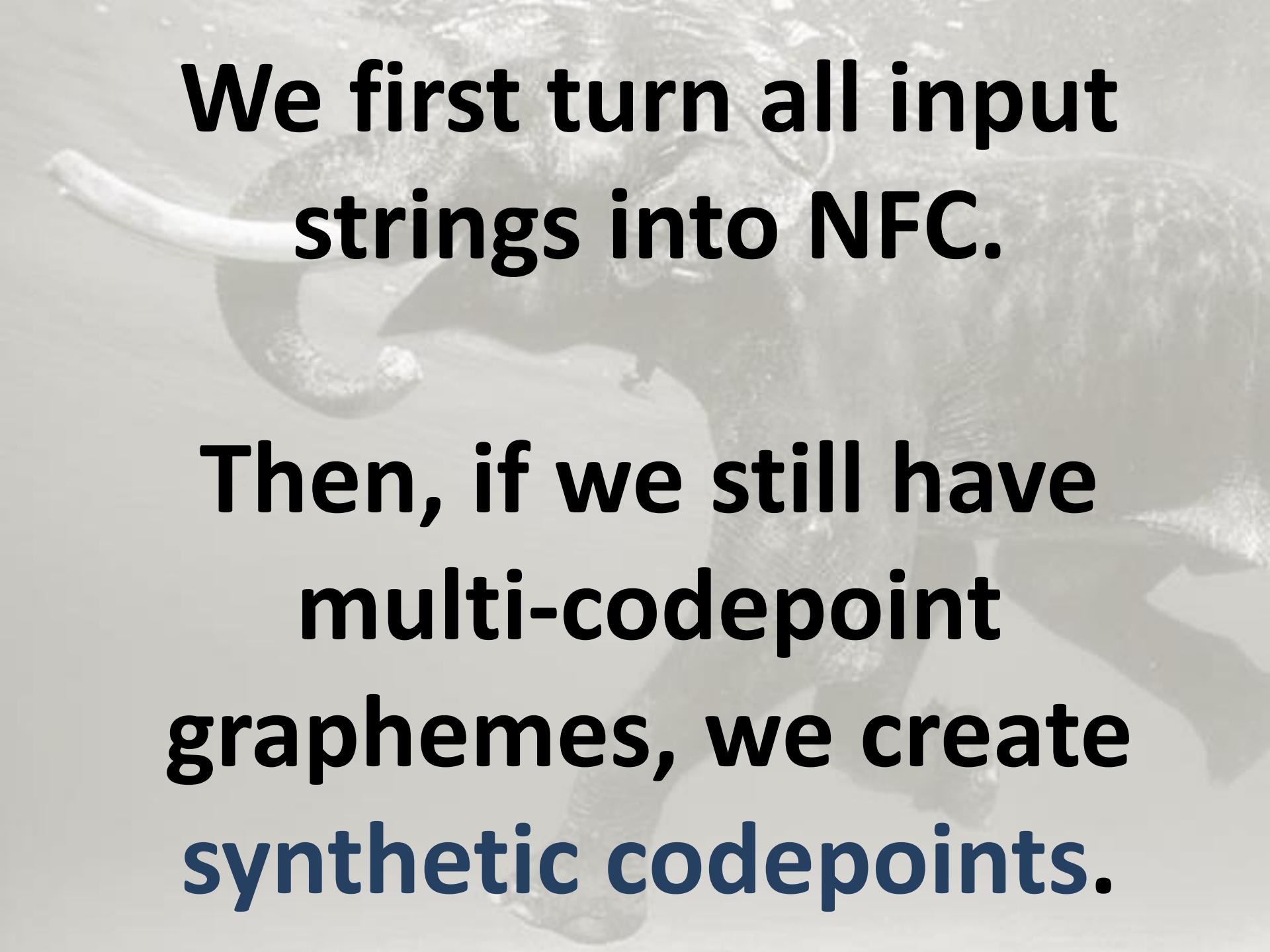


And...we get to have  $O(1)$   
string indexing.



A black and white photograph showing a person from the waist up, wearing a hard hat and a high-visibility safety vest over a dark shirt. They are leaning over, working on a large, textured industrial pipe or valve. The background is blurred, suggesting a workshop or factory environment.

# How do we do it?

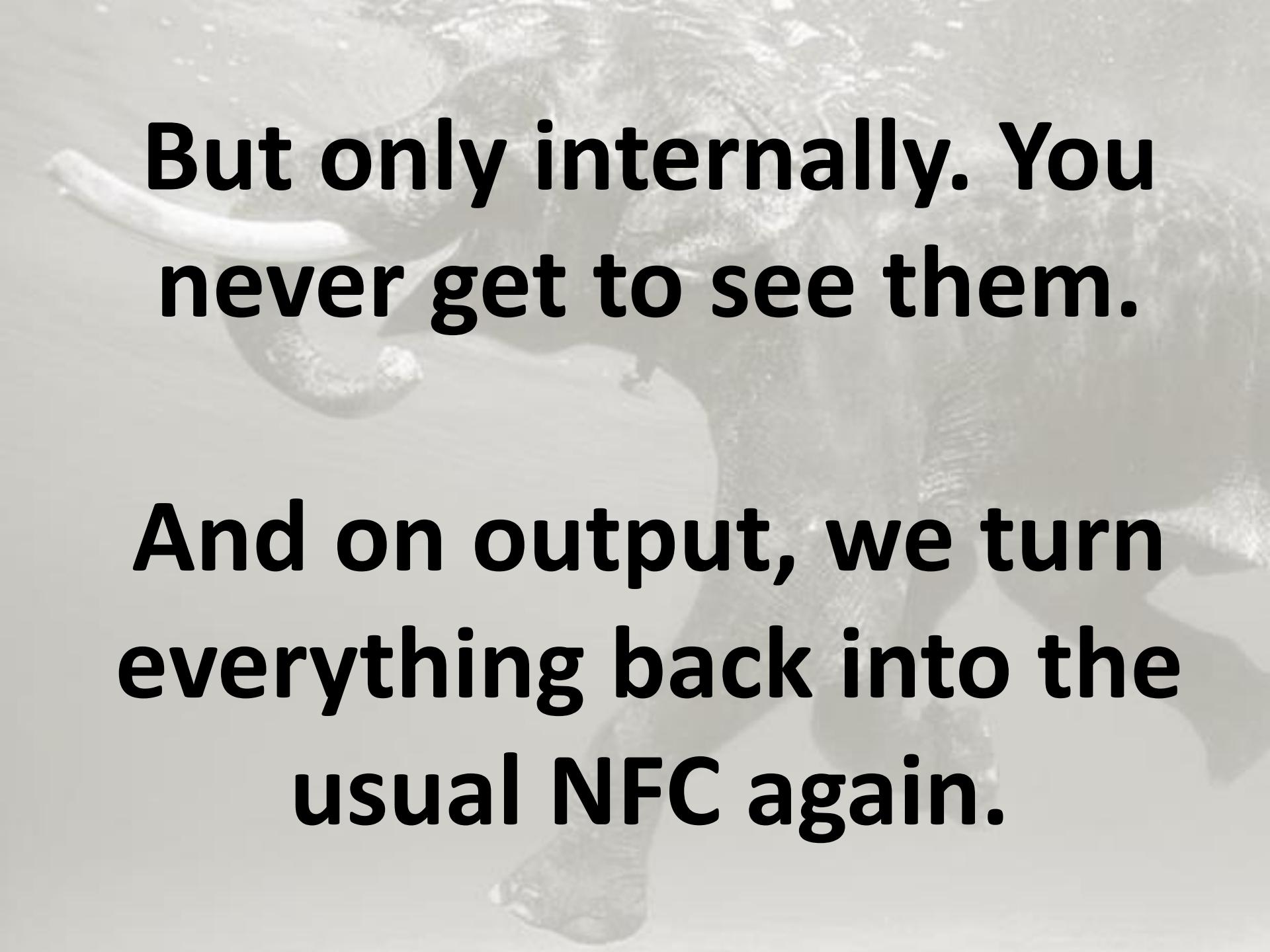


We first turn all input strings into NFC.

Then, if we still have multi-codepoint graphemes, we create synthetic codepoints.

Đ	a	t		r	o	c	k	s	!
-1	97	116	32	114	111	99	107	115	33

We represent synthetics  
using negative integers.



**But only internally. You  
never get to see them.**

**And on output, we turn  
everything back into the  
usual NFC again.**



Awesome!

# Questions?

