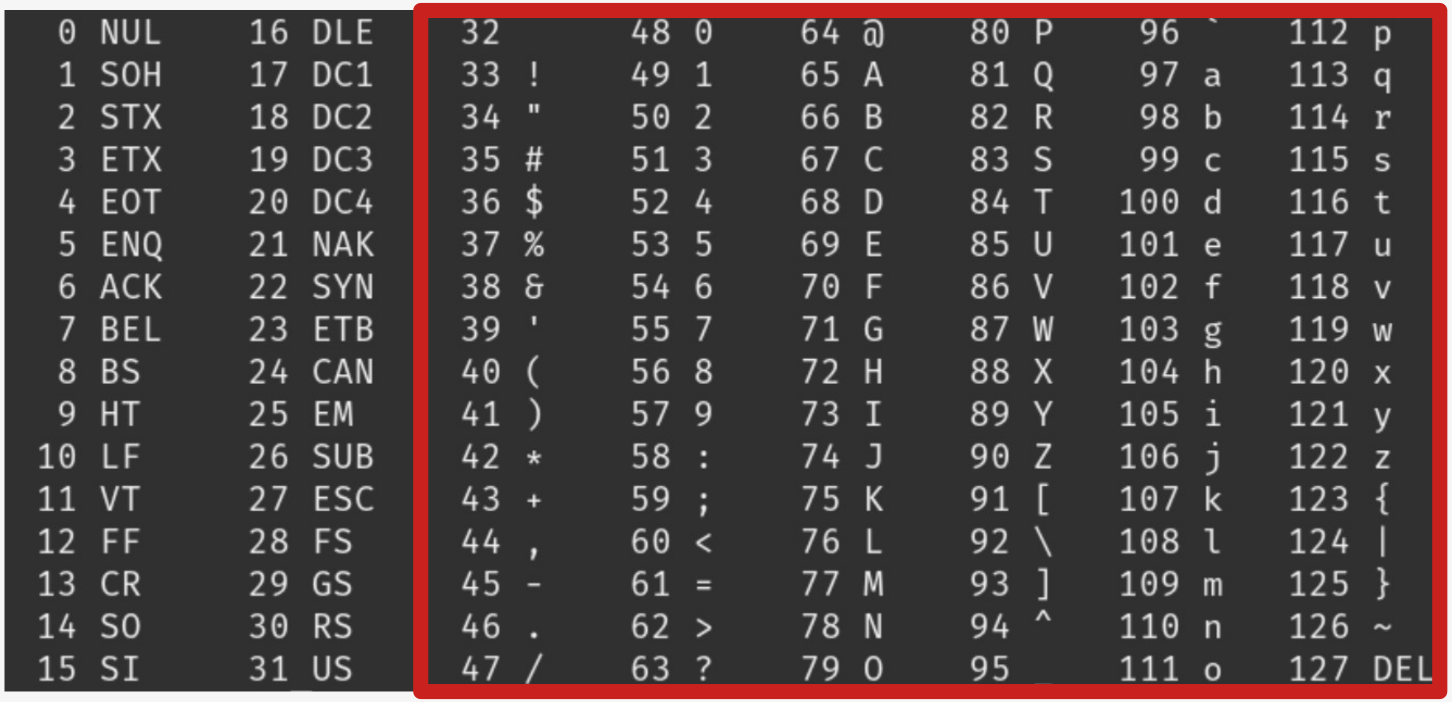
4-Byte Hell: When Unicode Enters the Stage

hack.lu Lightning Talk – Jonas Hess

24.10.2025

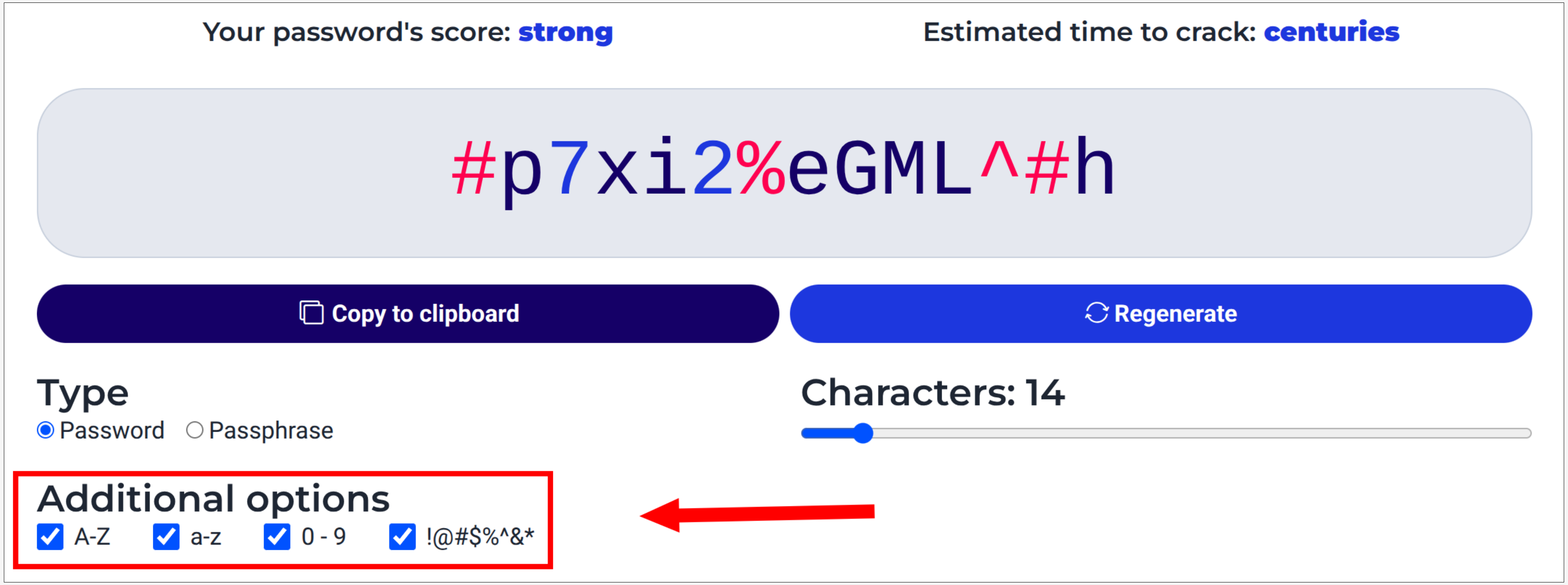
Typical Password Charset



```
  0 NUL    16 DLE    32        48 0    64 @    80 P     96 `    112 p
  1 SOH    17 DC1    33 !      49 1    65 A    81 Q     97 a    113 q
  2 STX    18 DC2    34 "      50 2    66 B    82 R     98 b    114 r
  3 ETX    19 DC3    35 #      51 3    67 C    83 S     99 c    115 s
  4 EOT    20 DC4    36 $      52 4    68 D    84 T    100 d    116 t
  5 ENQ    21 NAK    37 %      53 5    69 E    85 U    101 e    117 u
  6 ACK    22 SYN    38 &      54 6    70 F    86 V    102 f    118 v
  7 BEL    23 ETB    39 '      55 7    71 G    87 W    103 g    119 w
  8 BS     24 CAN    40 (      56 8    72 H    88 X    104 h    120 x
  9 HT     25 EM     41 )      57 9    73 I    89 Y    105 i    121 y
 10 LF     26 SUB    42 *      58 :    74 J    90 Z    106 j    122 z
 11 VT     27 ESC    43 +      59 ;    75 K    91 [    107 k    123 {
 12 FF     28 FS     44 ,      60 <    76 L    92 \    108 l    124 |
 13 CR     29 GS     45 -      61 =    77 M    93 ]    109 m    125 }
 14 SO     30 RS     46 .      62 >    78 N    94 ^    110 n    126 ~
 15 SI     31 US     47 /      63 ?    79 O    95 _    111 o    127 DEL
```

= 94 Chars

Password Managers encourage few Variability



Your password's score: **strong**

Estimated time to crack: **centuries**

```
#p7xi2%eGML^#h
```

Type

[✓ Password] Passphrase

Characters: 14

Additional options

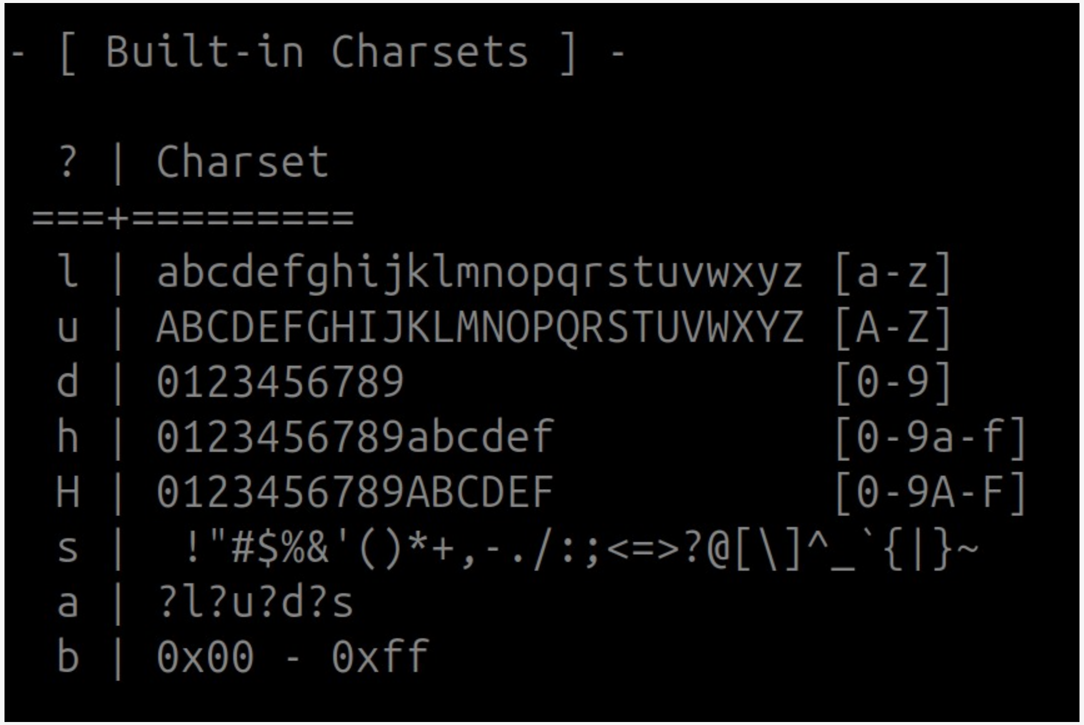
✓ A-Z ✓ a-z ✓ 0 - 9 ✓ !@#$%^&*

Mask Attacks in Hashcat

Example:

- Julia1984
- ?u?l?l?l?l?d?d?d?d

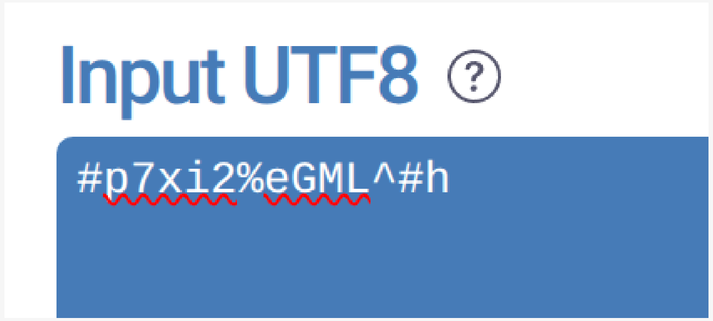
-?b?b?b?b?b?b?b?b?b



```
- [ Built-in Charsets ] -

  ? | Charset
 ===+=========
  l | abcdefghijklmnopqrstuvwxyz [a-z]
  u | ABCDEFGHIJKLMNOPQRSTUVWXYZ [A-Z]
  d | 0123456789                 [0-9]
  h | 0123456789abcdef           [0-9a-f]
  H | 0123456789ABCDEF           [0-9A-F]
  s |  !"#$%&'()*+,-./:;<=>?@[\]^_`{|}~
  a | ?l?u?d?s
  b | 0x00 - 0xff
```

Chars to Bytes

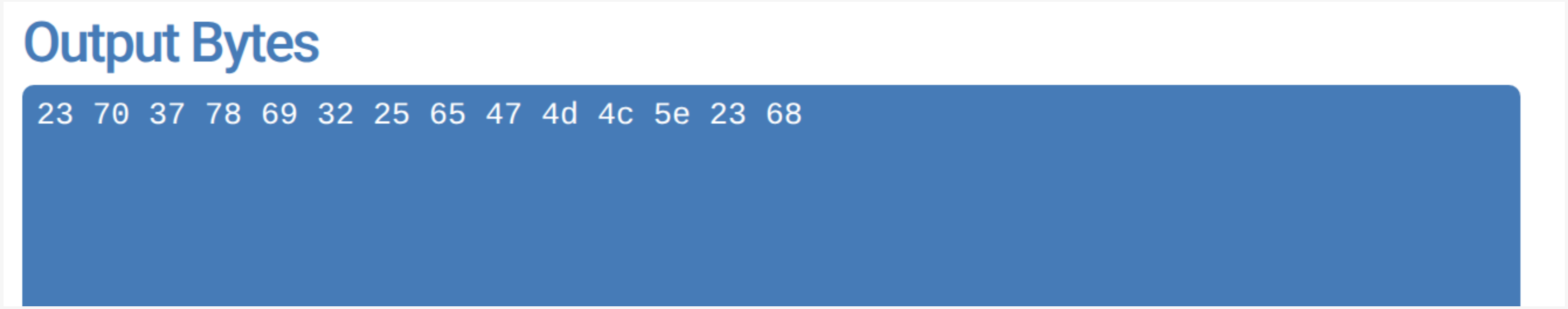


Input UTF8

```
#p7xi2%eGML^#h
```

14 chars → 14 Bytes

1:1 ratio

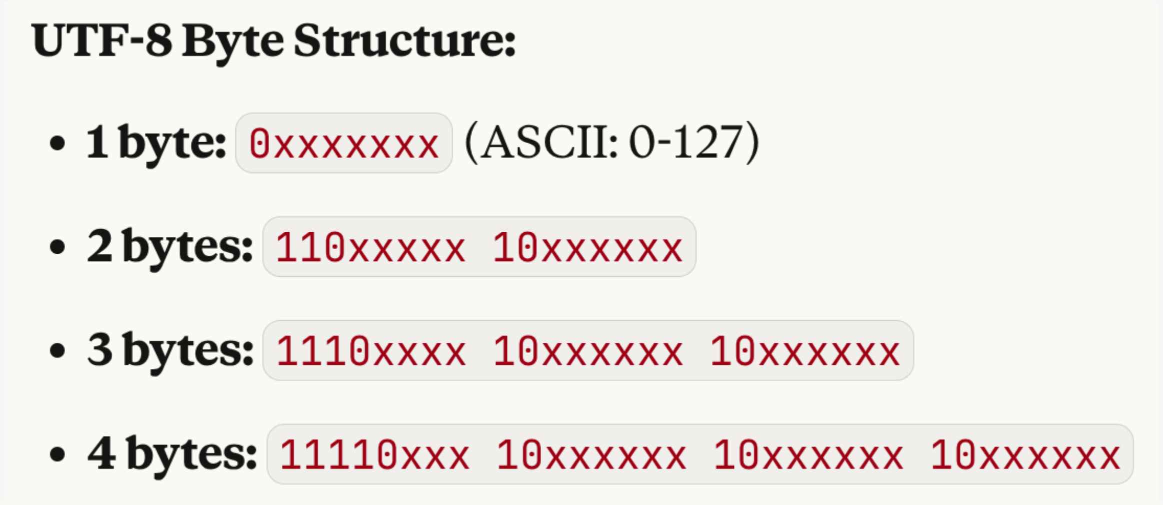


Output Bytes

```
23 70 37 78 69 32 25 65 47 4d 4c 5e 23 68
```

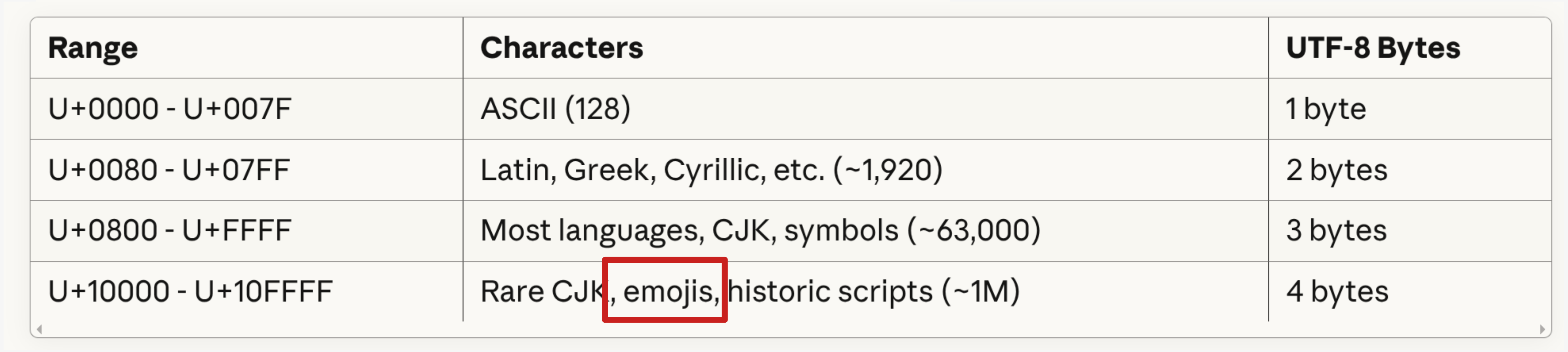
https://onlinetools.com/utf8/convert-utf8-to-bytes

Chars to Bytes



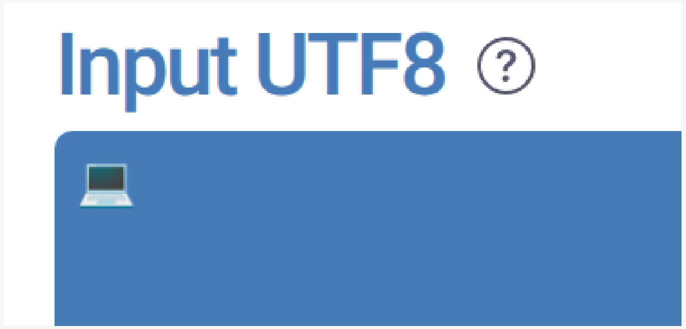
UTF-8 Byte Structure:

- **1 byte:** `0xxxxxxx` (ASCII: 0-127)
- **2 bytes:** `110xxxxx 10xxxxxx`
- **3 bytes:** `1110xxxx 10xxxxxx 10xxxxxx`
- **4 bytes:** `11110xxx 10xxxxxx 10xxxxxx 10xxxxxx`

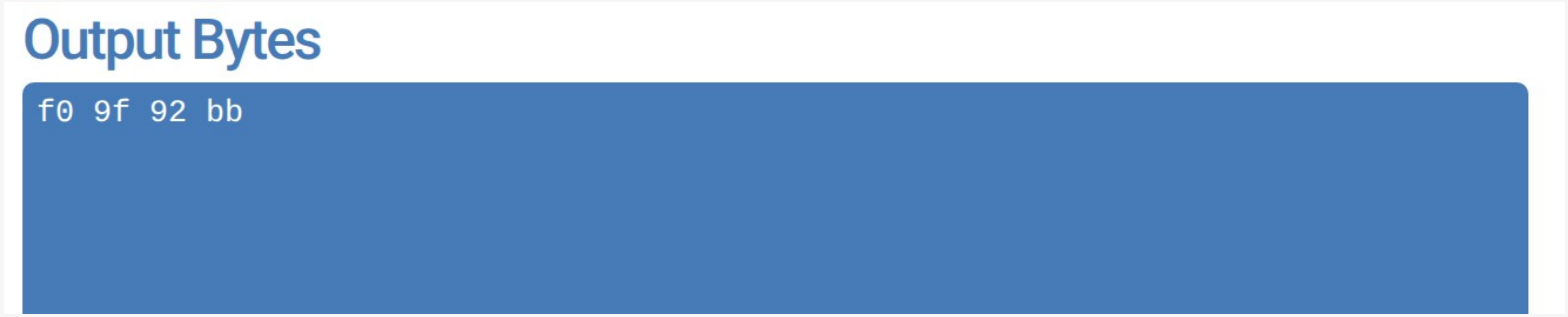


Range	Characters	UTF-8 Bytes
U+0000 - U+007F	ASCII (128)	1 byte
U+0080 - U+07FF	Latin, Greek, Cyrillic, etc. (~1,920)	2 bytes
U+0800 - U+FFFF	Most languages, CJK, symbols (~63,000)	3 bytes
U+10000 - U+10FFFF	Rare CJK, emojis, historic scripts (~1M)	4 bytes

Chars to Bytes


Input UTF8

💻

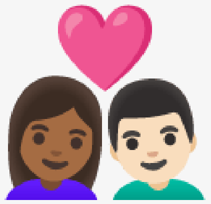

Output Bytes

```
f0 9f 92 bb
```

1 char → 4 Bytes

1:4 ratio

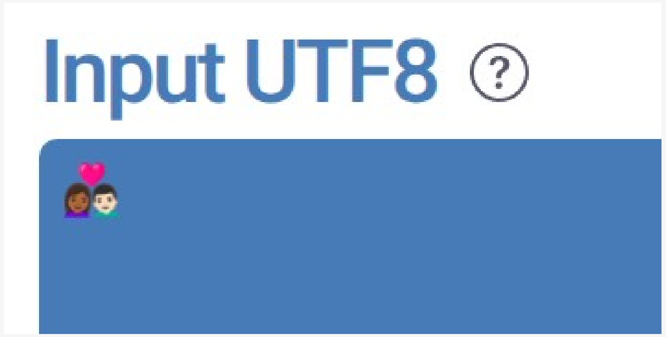
Complex Emojis



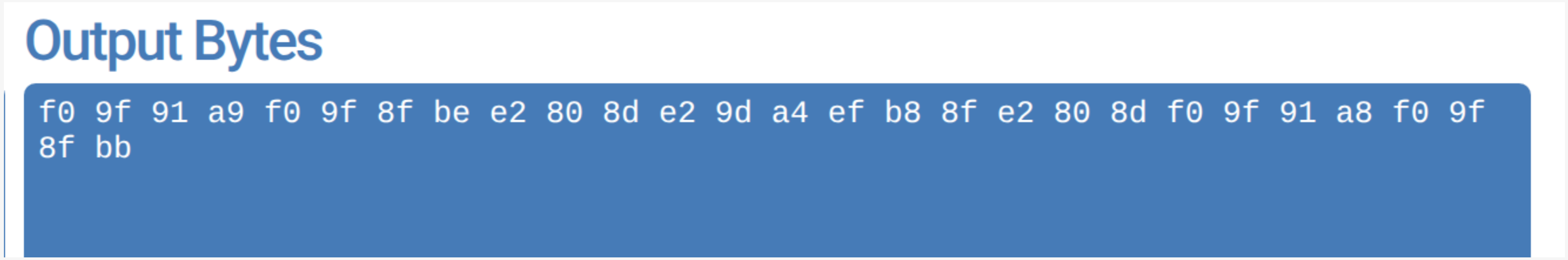
Couple with Heart and different Skin Tone

```
Component            Code Point     UTF-8 Bytes                 Byte Count
---------            ----------     -----------                 ----------
Woman                U+1F469        F0 9F 91 A9                 4 bytes
+ Skin (med-dark)    U+1F3FE        F0 9F 8F BE                 4 bytes
+ [glue]             U+200D         E2 80 8D                    3 bytes
+ Heart              U+2764         E2 9D A4                    3 bytes
+ [emoji style]      U+FE0F         EF B8 8F                    3 bytes
+ [glue]             U+200D         E2 80 8D                    3 bytes
+ Man                U+1F468        F0 9F 91 A8                 4 bytes
+ Skin (light)       U+1F3FB        F0 9F 8F BB                 4 bytes
                                    ──────────────────────────
                                    28 bytes total
```

Chars to Bytes – Quality over Quantity



👩🏾‍❤️‍👨🏻



```
f0 9f 91 a9 f0 9f 8f be e2 80 8d e2 9d a4 ef b8 8f e2 80 8d f0 9f 91 a8 f0 9f
8f bb
```

1 char → 28 Bytes

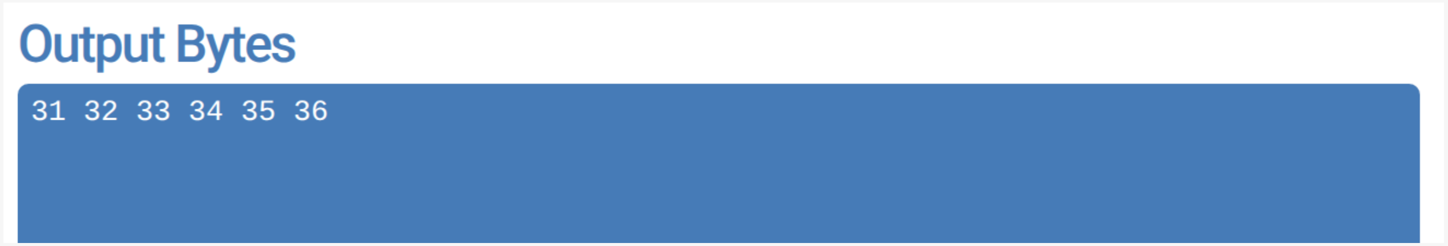
1:28 ratio

Back to the Roots

Instead of:

6 Bytes

123456



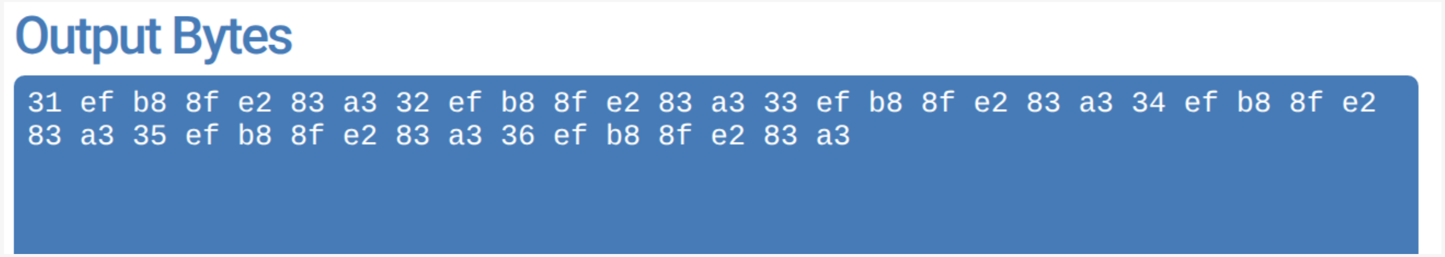
Output Bytes

```
31 32 33 34 35 36
```

Think:

1️⃣2️⃣3️⃣4️⃣5️⃣6️⃣

36 Bytes



Output Bytes

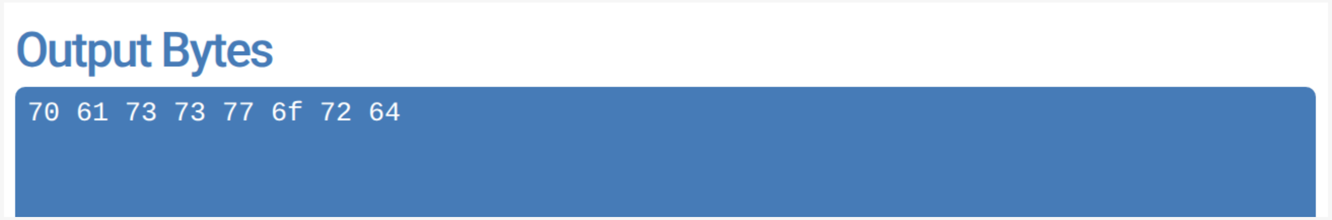
```
31 ef b8 8f e2 83 a3 32 ef b8 8f e2 83 a3 33 ef b8 8f e2 83 a3 34 ef b8 8f e2
83 a3 35 ef b8 8f e2 83 a3 36 ef b8 8f e2 83 a3
```

Back to the Roots

Instead of:

password

8 Bytes

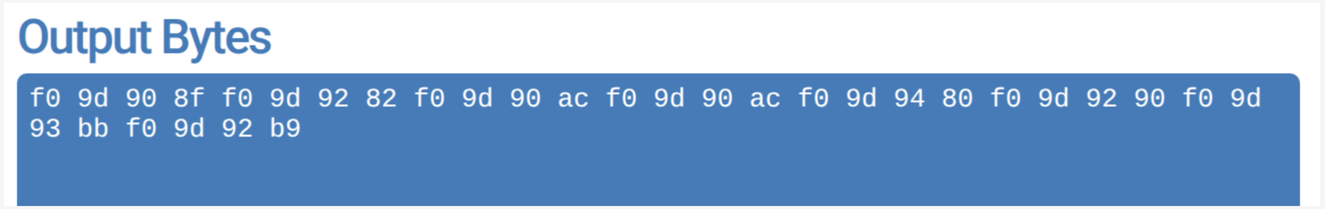

Output Bytes

```
70 61 73 73 77 6f 72 64
```

Think:

𝐏𝒂𝐬𝐬𝔀𝒐𝓻𝒹

32 Bytes

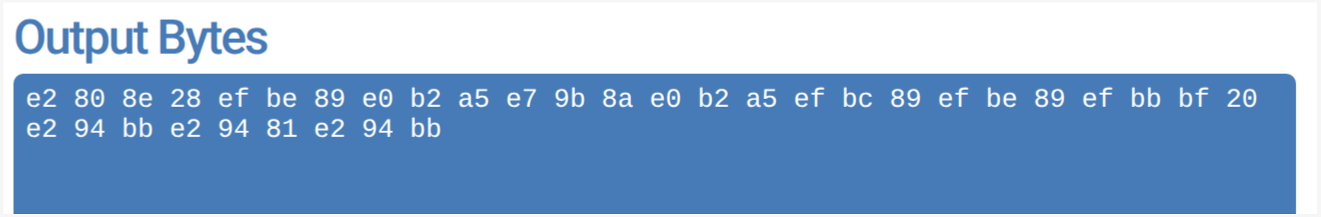

Output Bytes

```
f0 9d 90 8f f0 9d 92 82 f0 9d 90 ac f0 9d 90 ac f0 9d 94 80 f0 9d 92 90 f0 9d
93 bb f0 9d 92 b9
```

Get Creative

(ﾉಥ益ಥ）ﾉ ┻━┻

35 Bytes

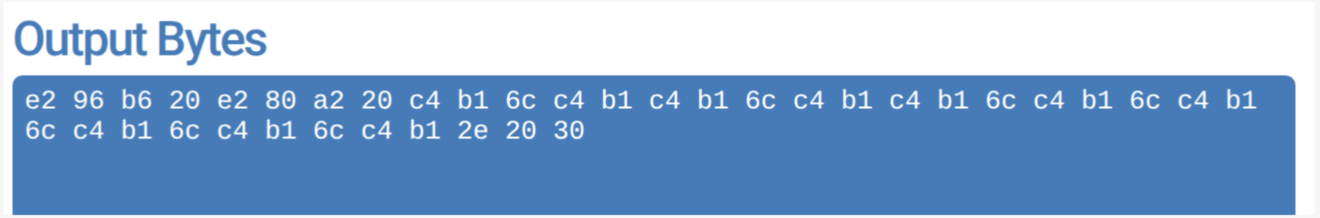


Output Bytes

```
e2 80 8e 28 ef be 89 e0 b2 a5 e7 9b 8a e0 b2 a5 ef bc 89 ef be 89 ef bb bf 20
e2 94 bb e2 94 81 e2 94 bb
```

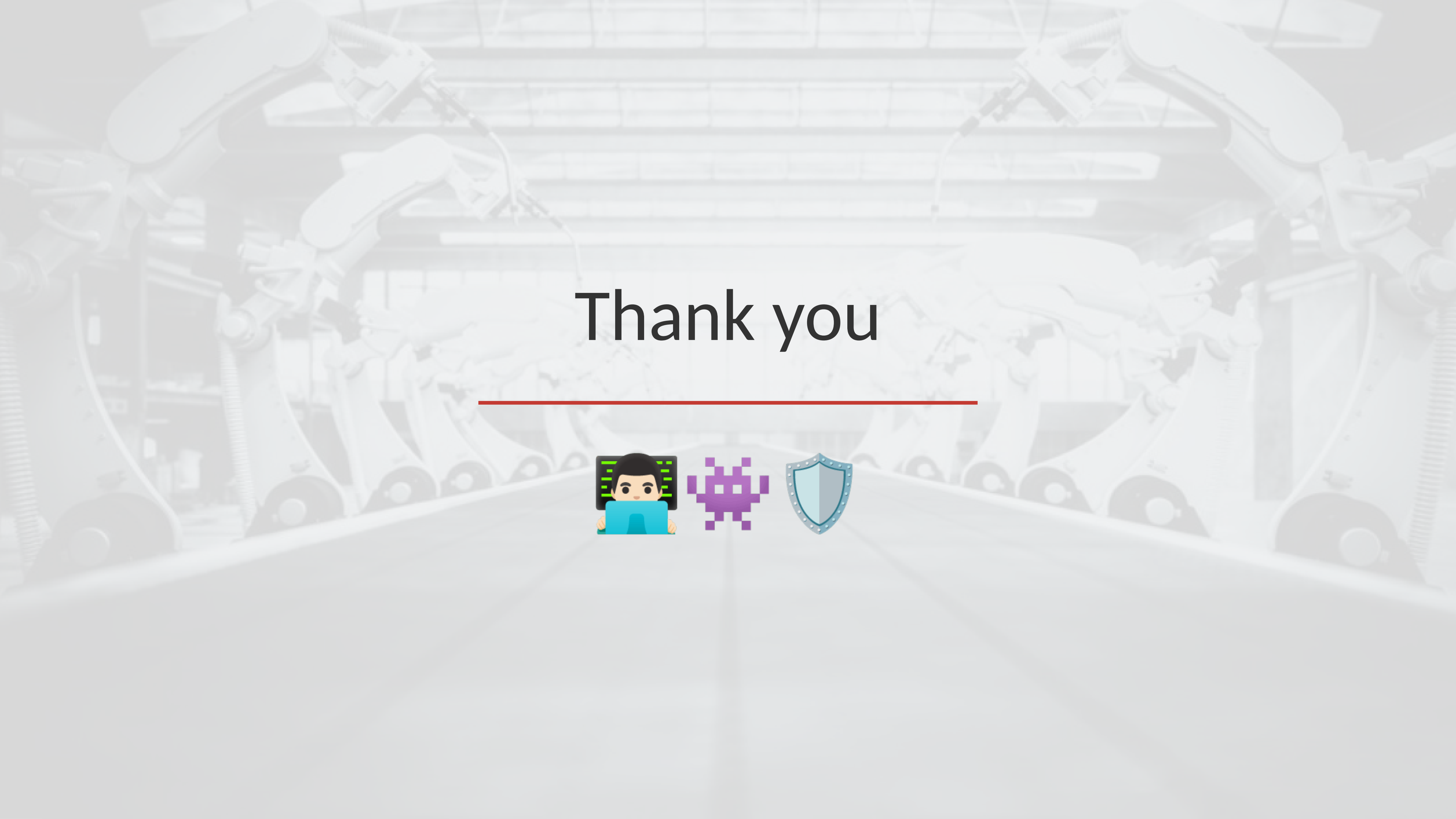
▶️ • ılıılıılılılılılı. 0

38 Bytes



Output Bytes

```
e2 96 b6 20 e2 80 a2 20 c4 b1 6c c4 b1 c4 b1 6c c4 b1 c4 b1 6c c4 b1 6c c4 b1
6c c4 b1 6c c4 b1 6c c4 b1 2e 20 30
```



Thank you

👨🏻‍💻 👾 🛡️