l = list() ⟶ l = [ ]

d = dict() ⟶ d = { }

t = tuple() ⟶ t = ( )

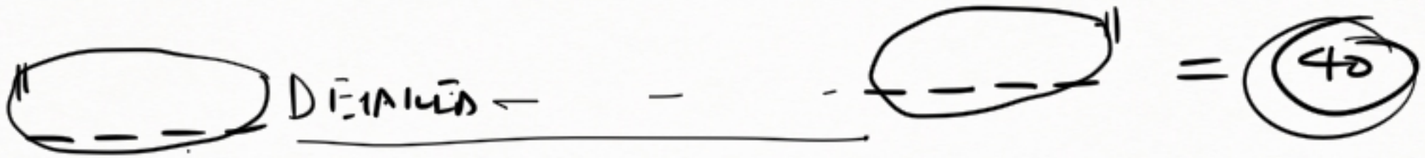
s = set() ⟶ s = set( )

s = str() ⟶ s = " "

SYNTACTIC SUGAR

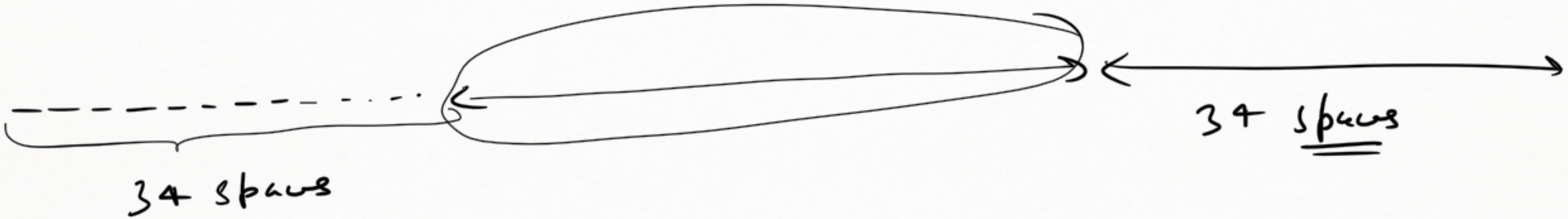
40

S = "DETAILED_INTRODUCTION_TO_STRINGS"

len(S) = 

= 40

100 - 32 ⇒ 68



BASE-10
BASE-2
BASE-16
} NUMBER SYSTEMS

Base-2 → Total 2 number of unique digits available for counting

BASE-x → Total x number of digits (unique) available for counting

Base-10 → Total 10 number of unique digits available for counting

Base-16 → You total 16 number of digits available for Counting

Base-10 → 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

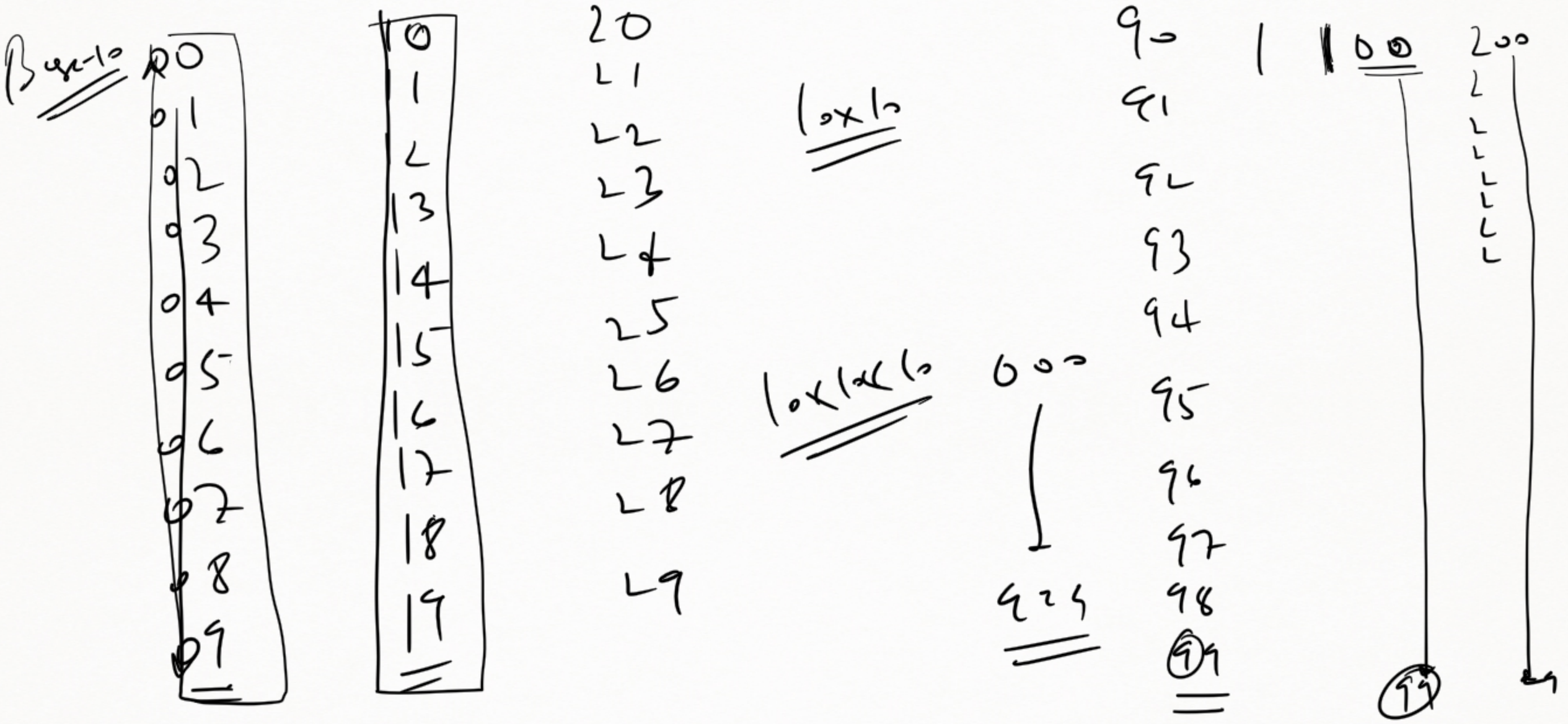
1, 30, 95, 701

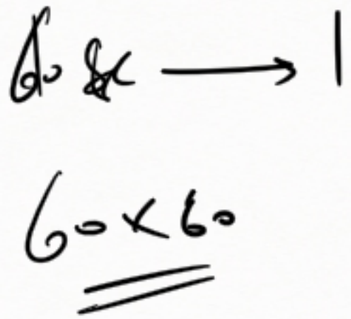
Base-2 :- 0, 1

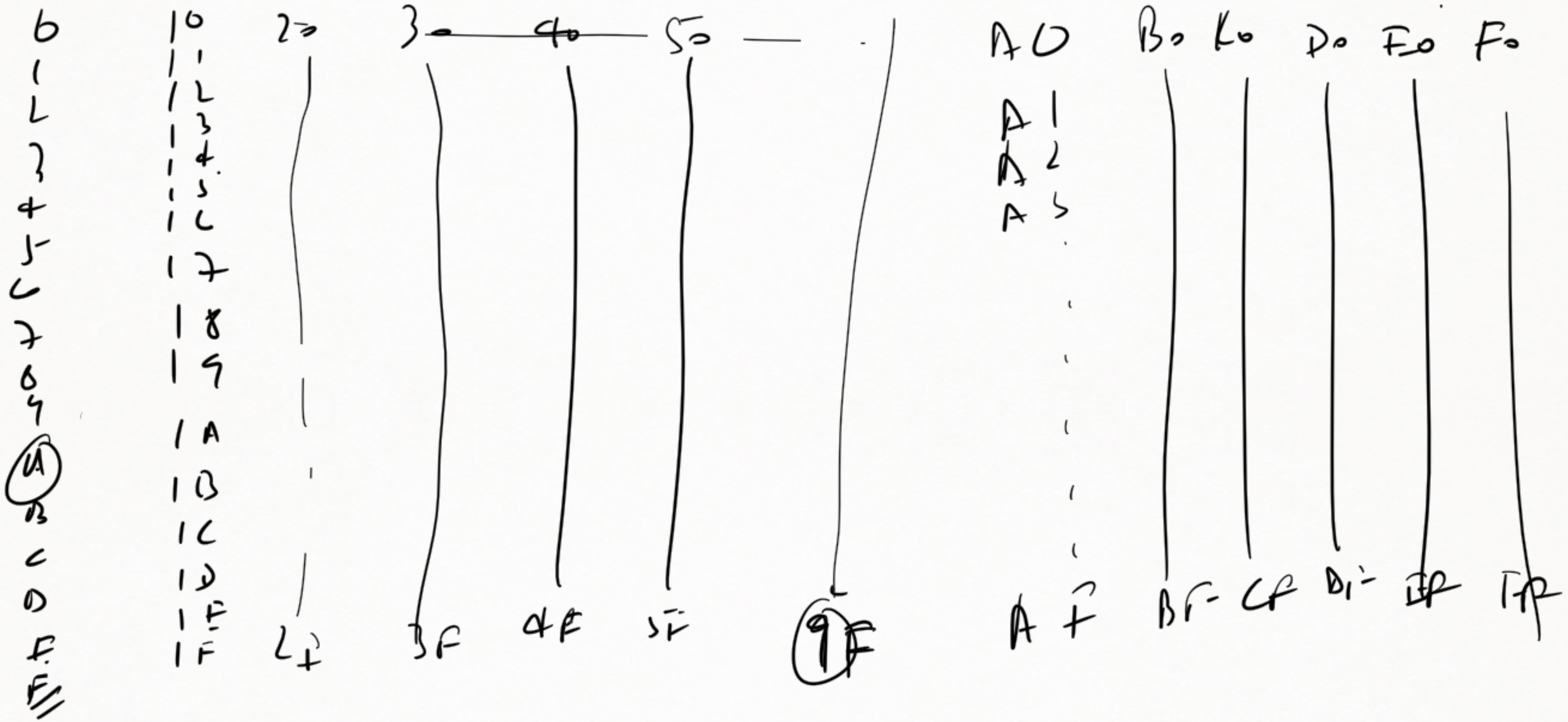
10100100100011

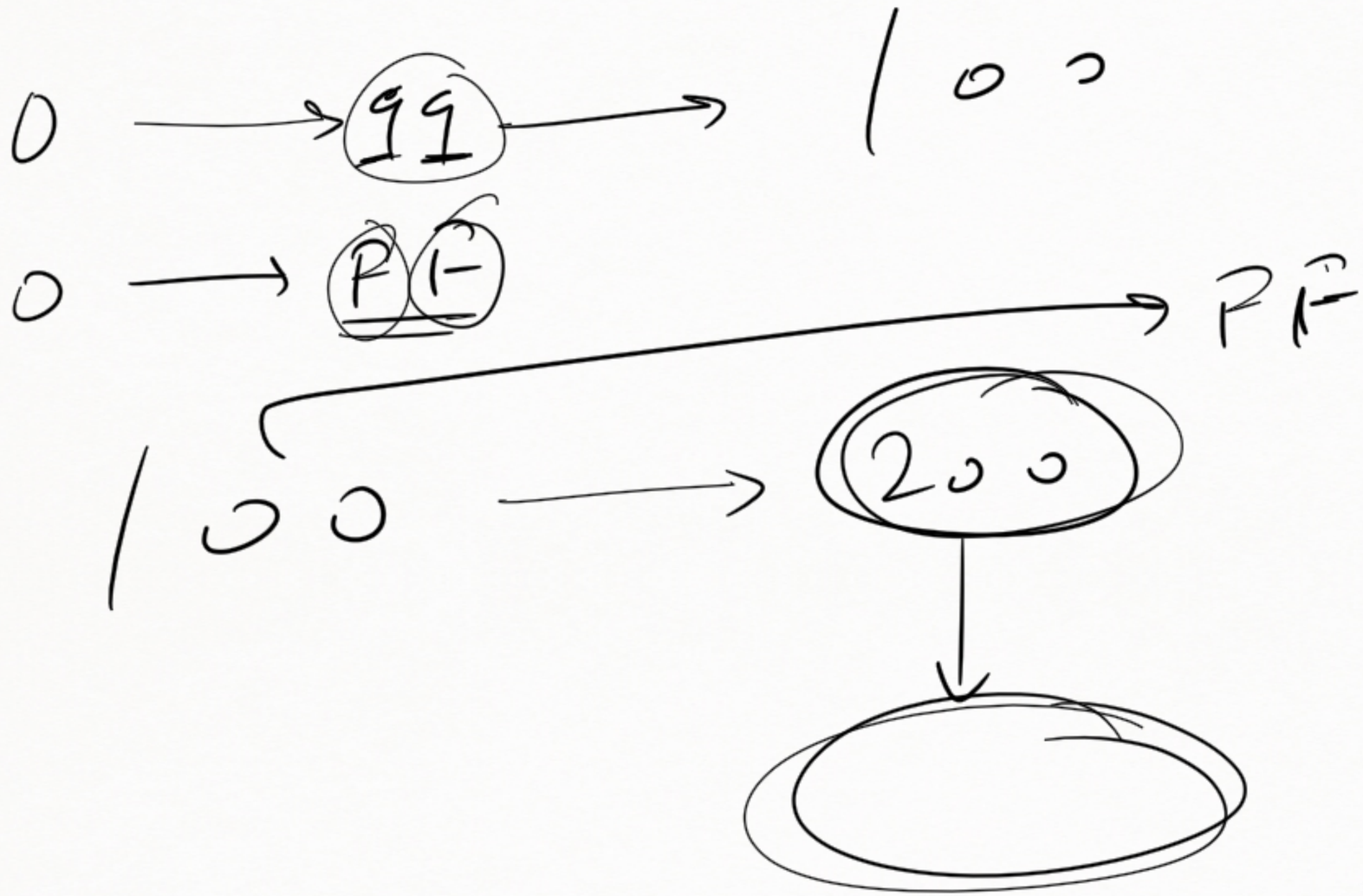
Base-16 :- 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

FQCABD7C ⟶ Base 16 Number


Base-10
10x10
10x10x10
100







1FF
200
2FF
300
3FF
400

1000

FFF

1FFF

2000

2FFF

3000

1FF

200

2FF

300

9FF

A00

AFF

B00

BFF

C00

CFF

D00

DFF

E00

EFF

F00

FFF

1000

11FF

3000

3FFF

4000

4FFF

5000

5FFF

9000

9FFF

A000

AFFF

B000

BFFF

C000

CFFF

D000

DFFF

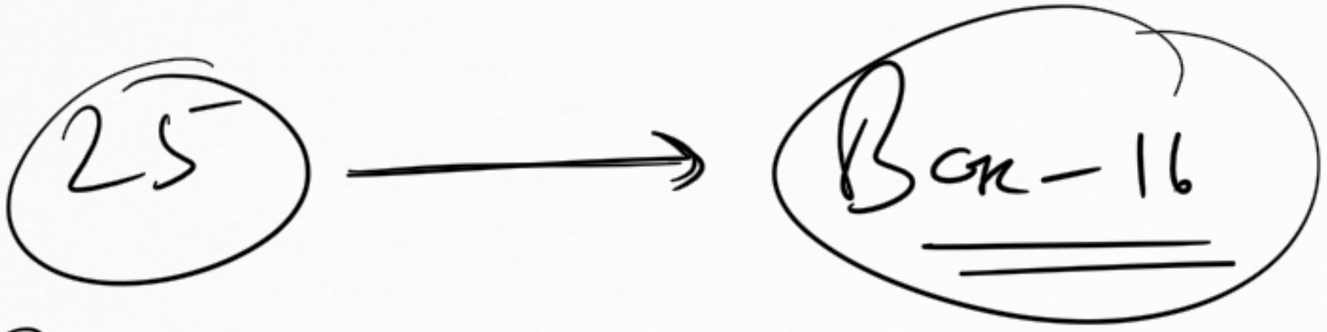
E000

EFFF

F000

FFFF

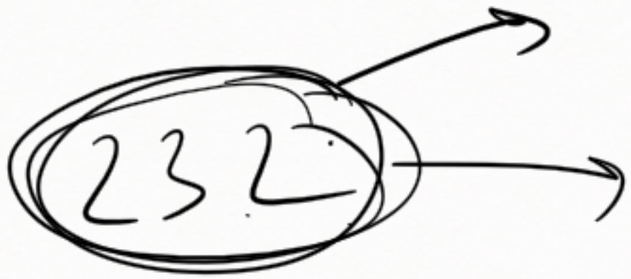
10000



Box-10

0-15

0 → 0
1 → 1
2 → 2
3 → 3
4 → 4
5 → 5
6 → 6
7 → 7
8 → 8
9 → 9
10 → A
11 → B
12 → C
13 → D
14 → E
15 → F
16 → 10
17 → 11
18 → 12
19 → 13
20 → 14
21 → 15
22 → 16
23 → 17
24 → 18
25 → 19



1 min = 60 kB

1 sec = 16 counts

2 sec = 32

3 sec = 48

4 sec = 64

80

96

112

128

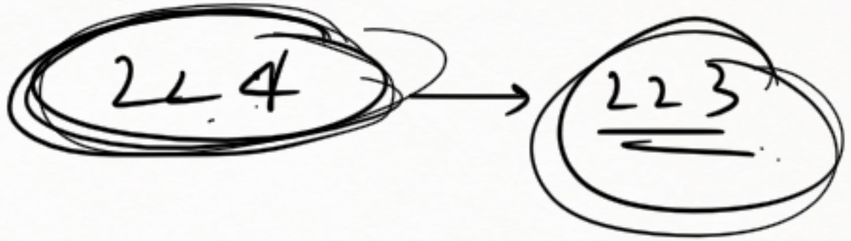
144

160

176

192

208

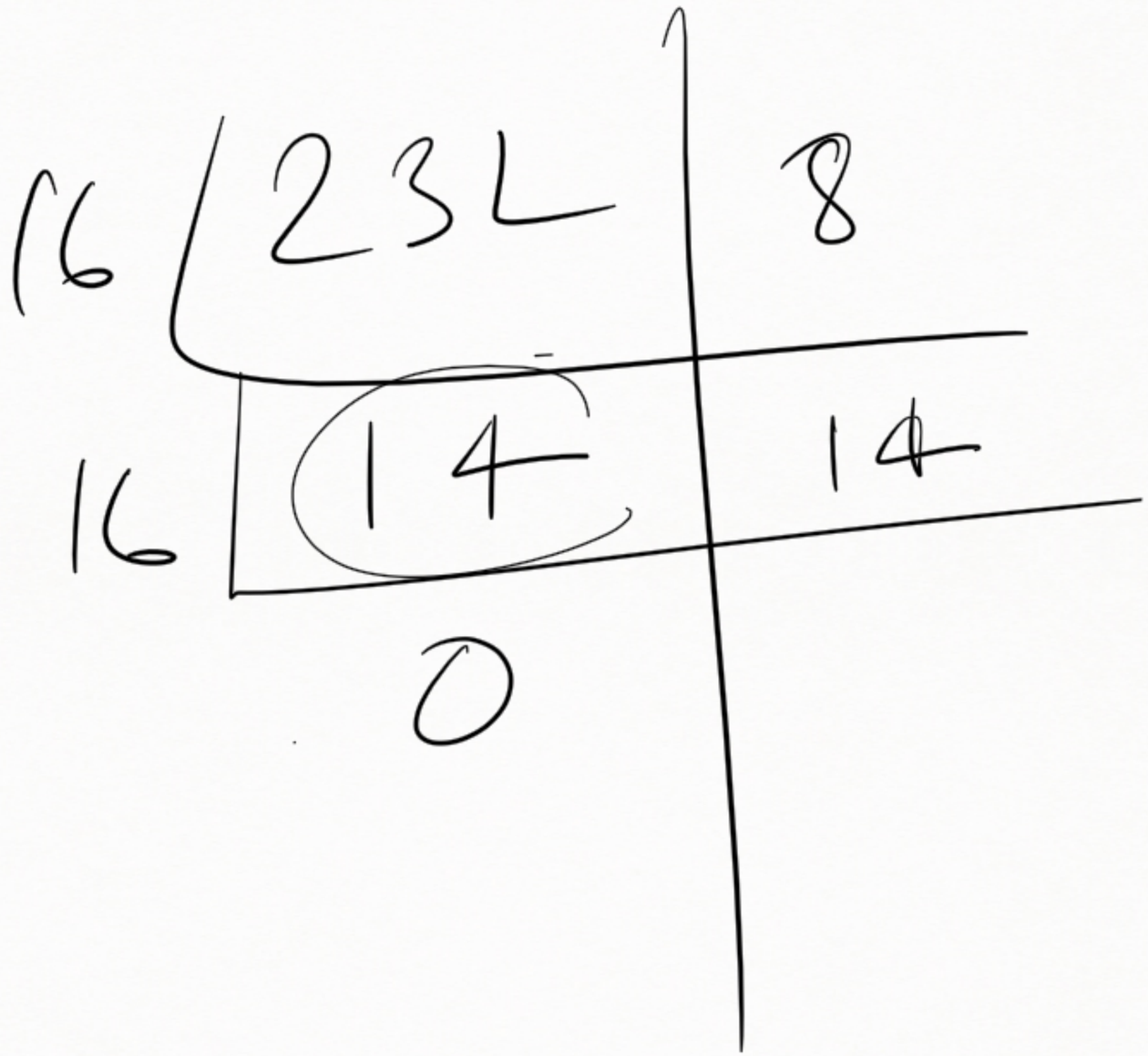


E9

16 x 14

224

232
-224
8



9

153 − 0

9

16 × 9 = 144

(99)

153

−144

9

&

	153	R
16		9
16	9	9
	0	

99

335

335

15 → F

16 × 14

16 × 18 → 288

16 × 19 → 304

16 × 20 → 320

335

0 ← F → 15

1

2

3

4

5

6

7

8

9 ← 9

10 ← A

11 ← B

12 ← C

13 ← D

E → 14

15 → F

16 → 10

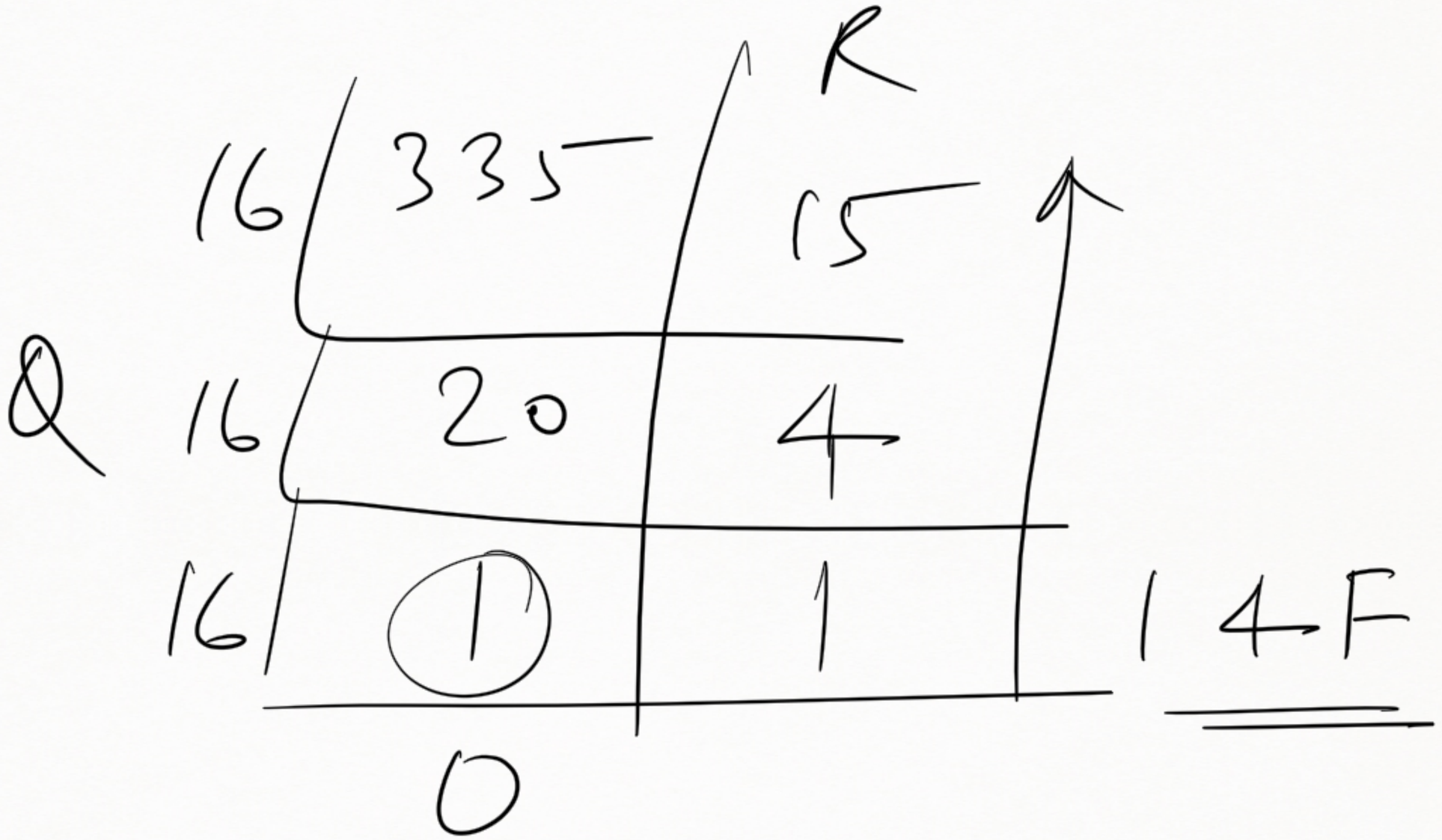
17 → 11

18 → 12

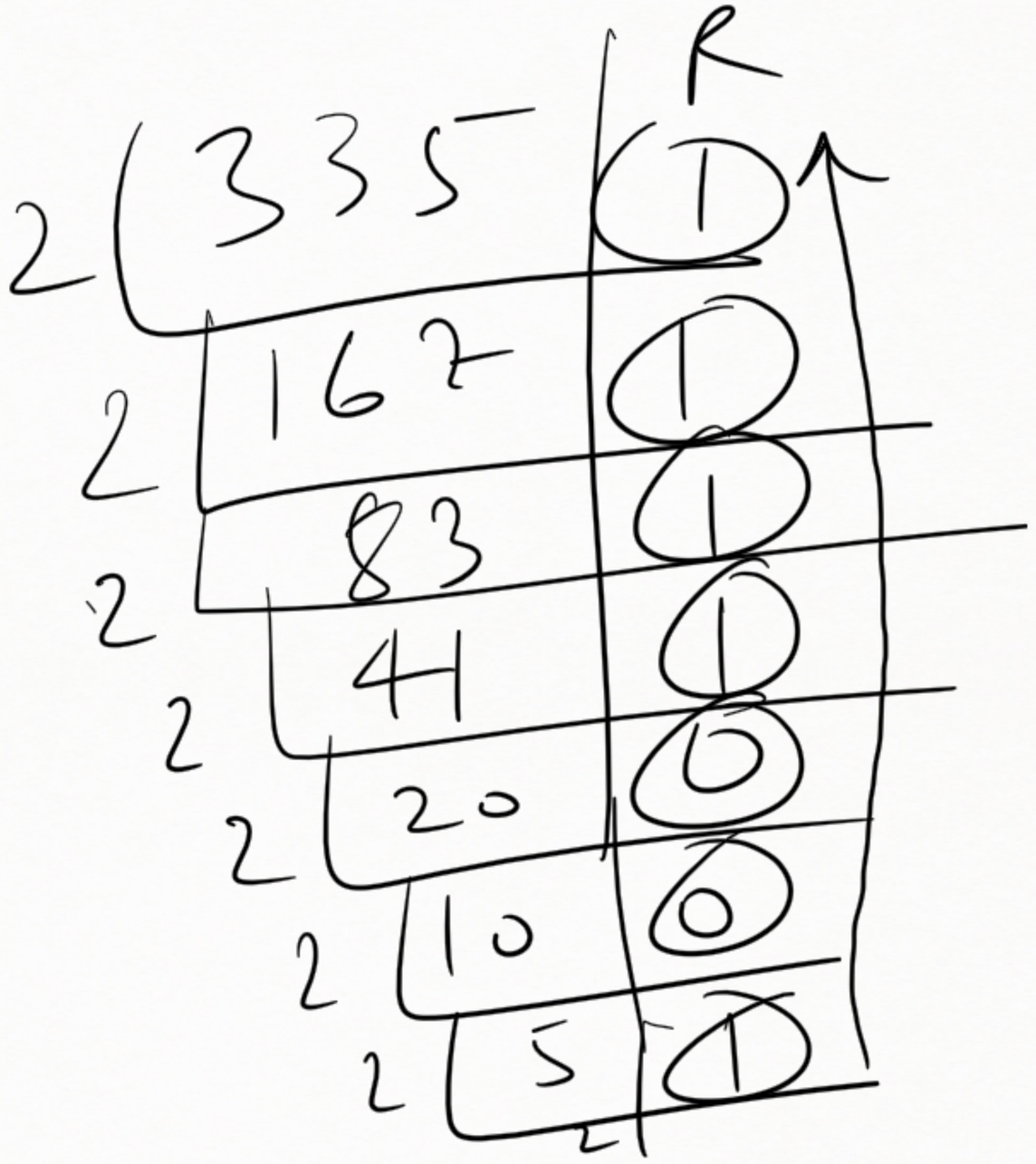
19 → 13

20 → 14

14F


R
16
335
15
Q
16
20
4
16
1
1
14F
0

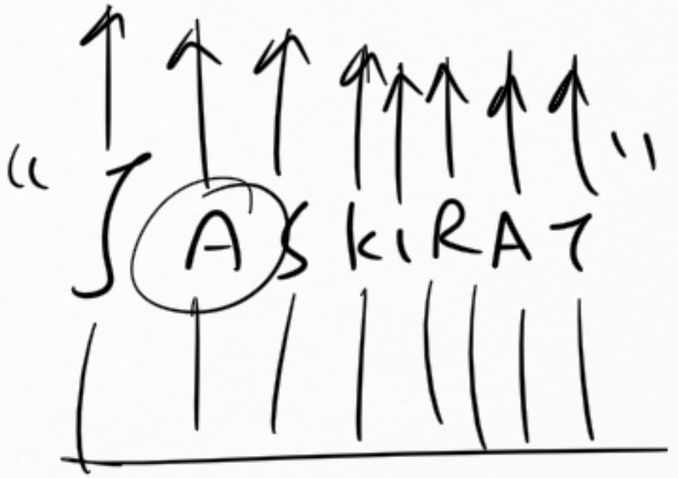
335



101001111

Strings ⟶ Encode ASCII

"Jaskirat" ⟶ Set of numbers



20

Encoding Schemes

UTF-8

LATIN ASCII Codes

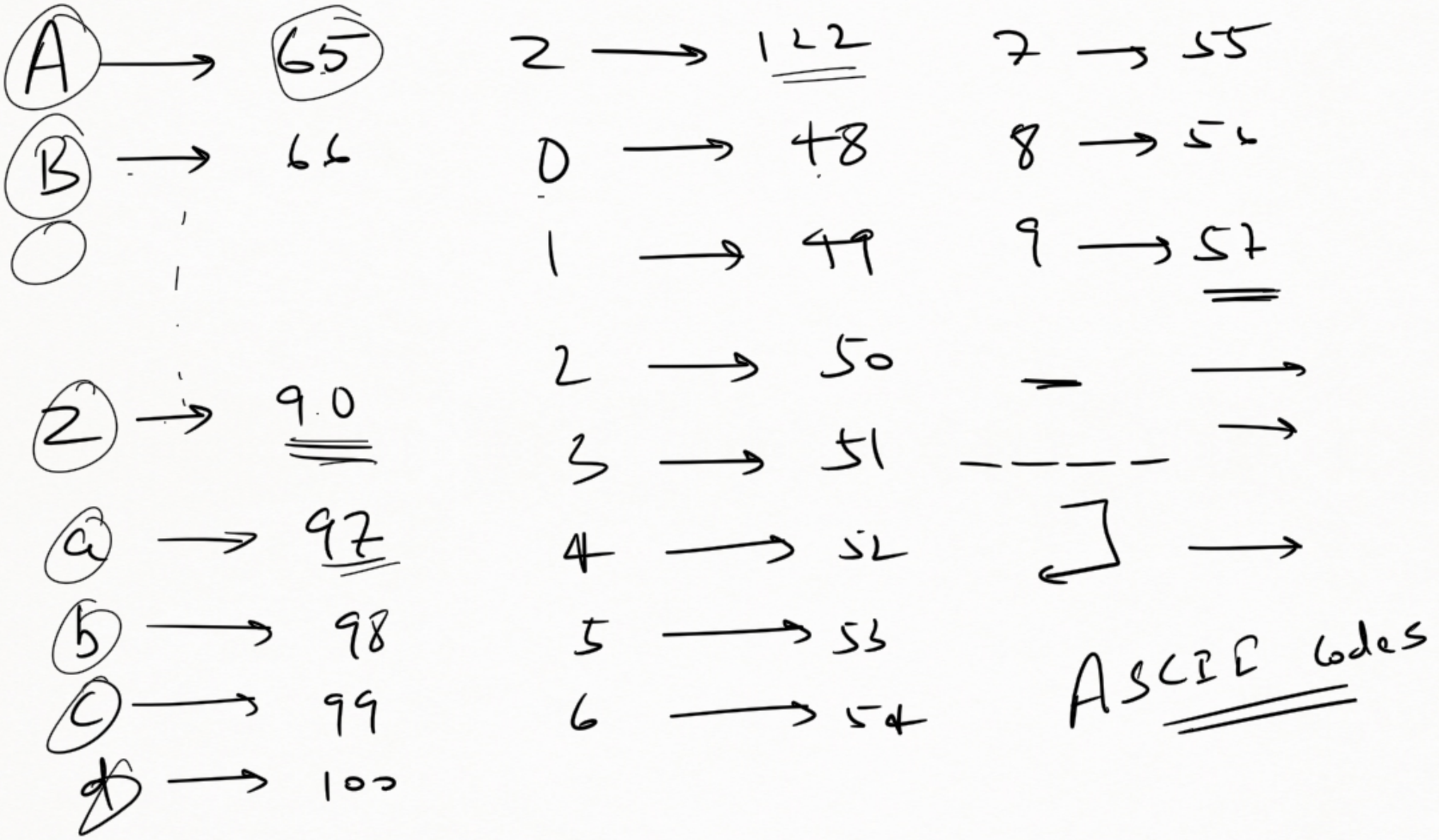
UTF-8, ASCII, LATIN, ---- 20

ASCII → AMERICAN STANDARD CODE FOR

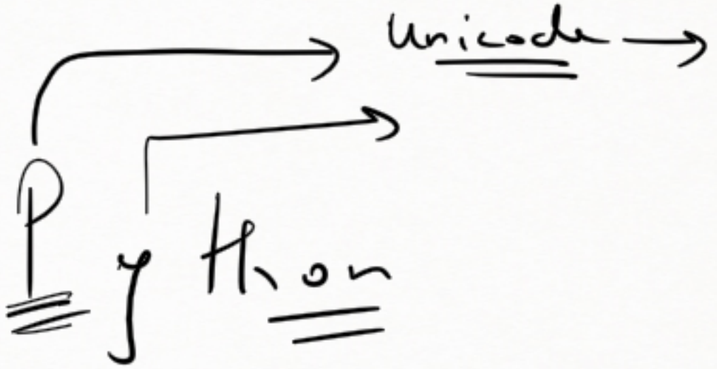
INFORMATION INTERCHANGE

128 Characters → Encode

!, /, +, =, @, $, A, B, C, D, E, ... Z, 0, 1, 2, ... 9, a, b, c, ... z


A → 65
B → 66
Z → 90
a → 97
b → 98
c → 99
d → 100
2 → 122
0 → 48
1 → 49
2 → 50
3 → 51
4 → 52
5 → 53
6 → 54
7 → 55
8 → 56
9 → 57
ASCII codes

UTF-8 :



→ ASCII

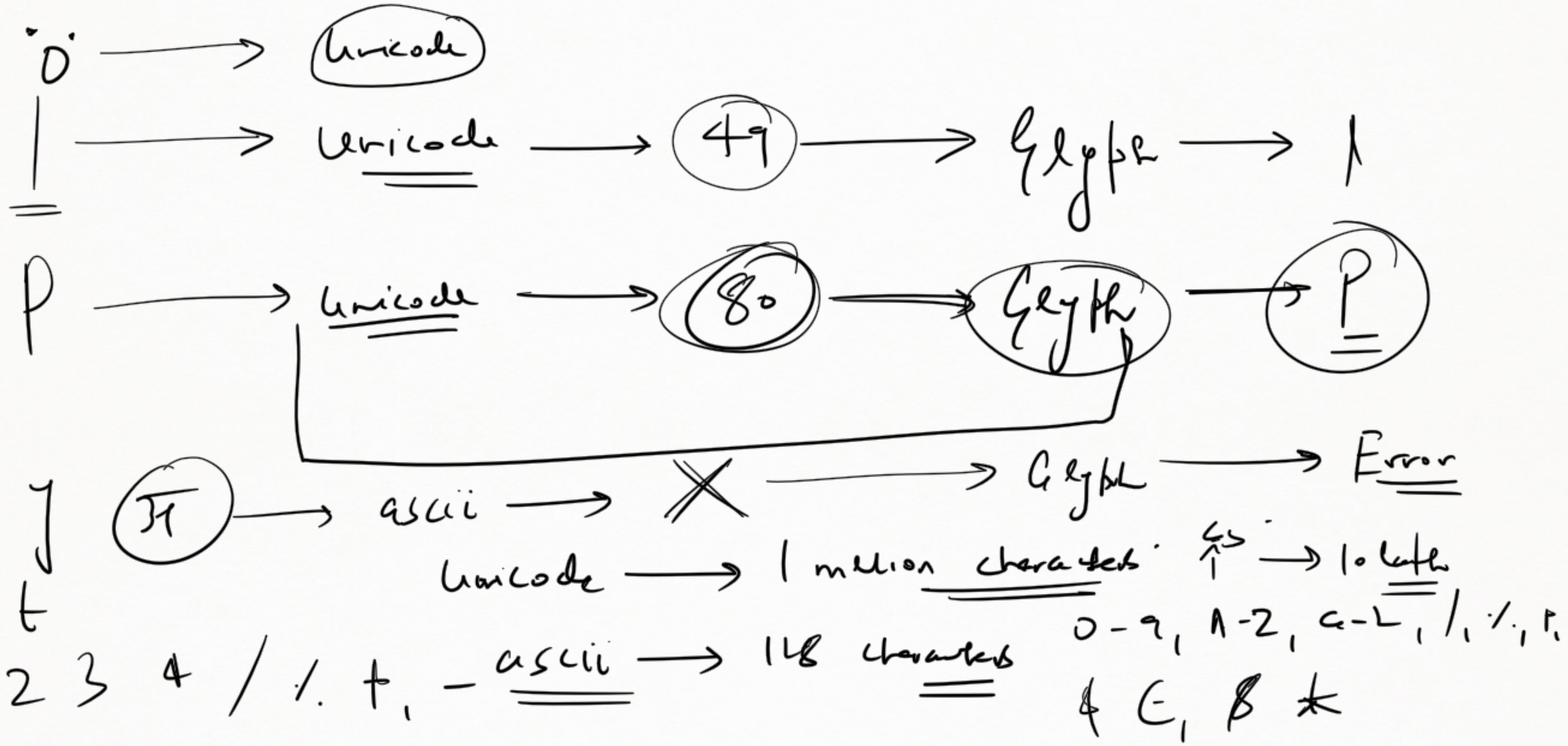
→ Character doesn't come under ASCII

"Python".encode("ascii")

ascii → 0 – 127 →

Python

1234


'0'
Unicode
Unicode
49
Glyph
1
p
Unicode
80
Glyph
P
ascii
Glyph
Error
Unicode
1 million characters
10 bits
0-9, A-Z, a-z, !, %, ?, -
2 3 4 / %. +, -
ascii
128 characters
& €, $ *

A B C D E F G H I J K

65 66 67 68 69 70 71 72 73 74 75

L M N O P

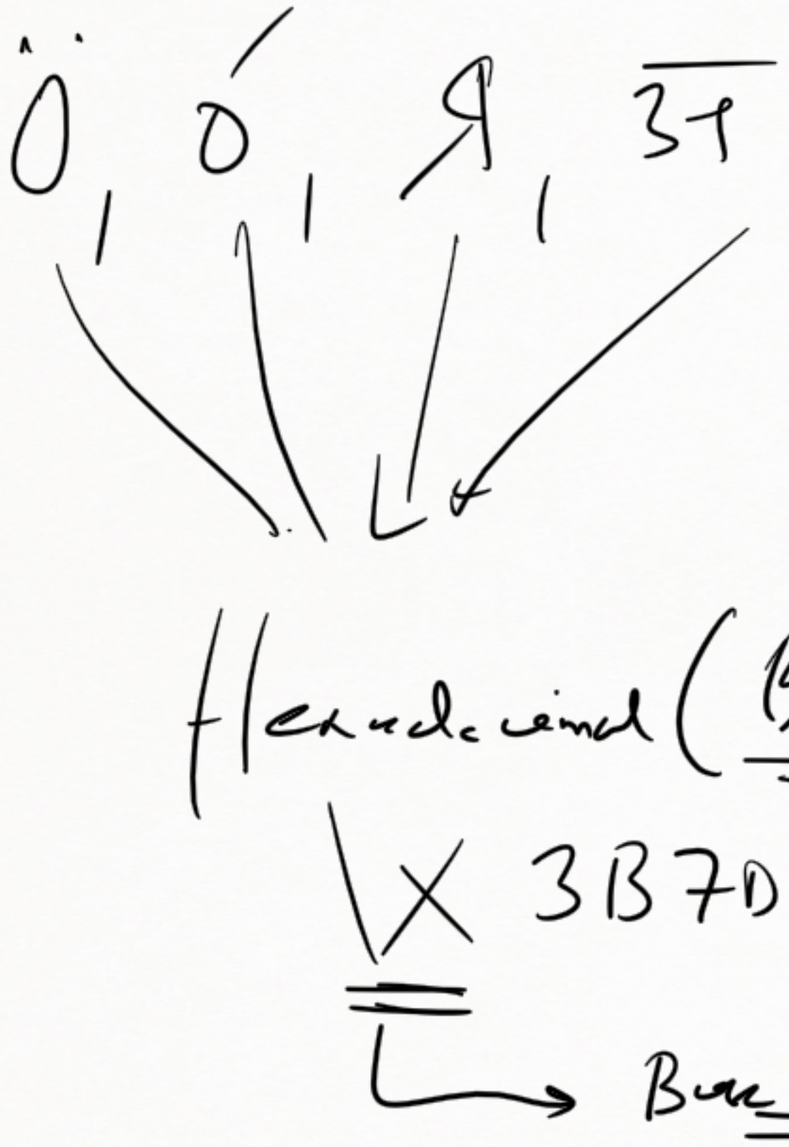
76 77 78 79 80

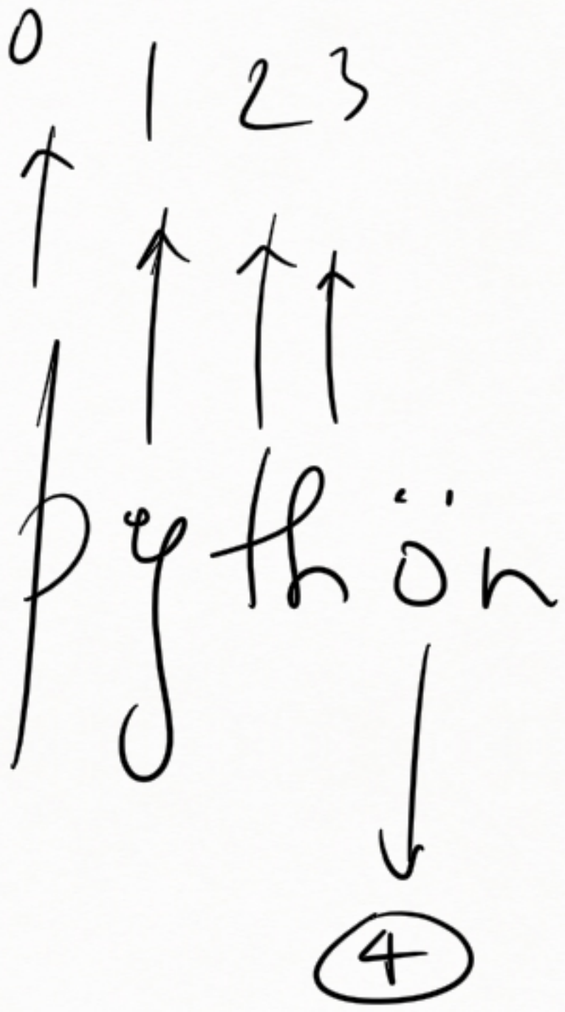
Ö → 

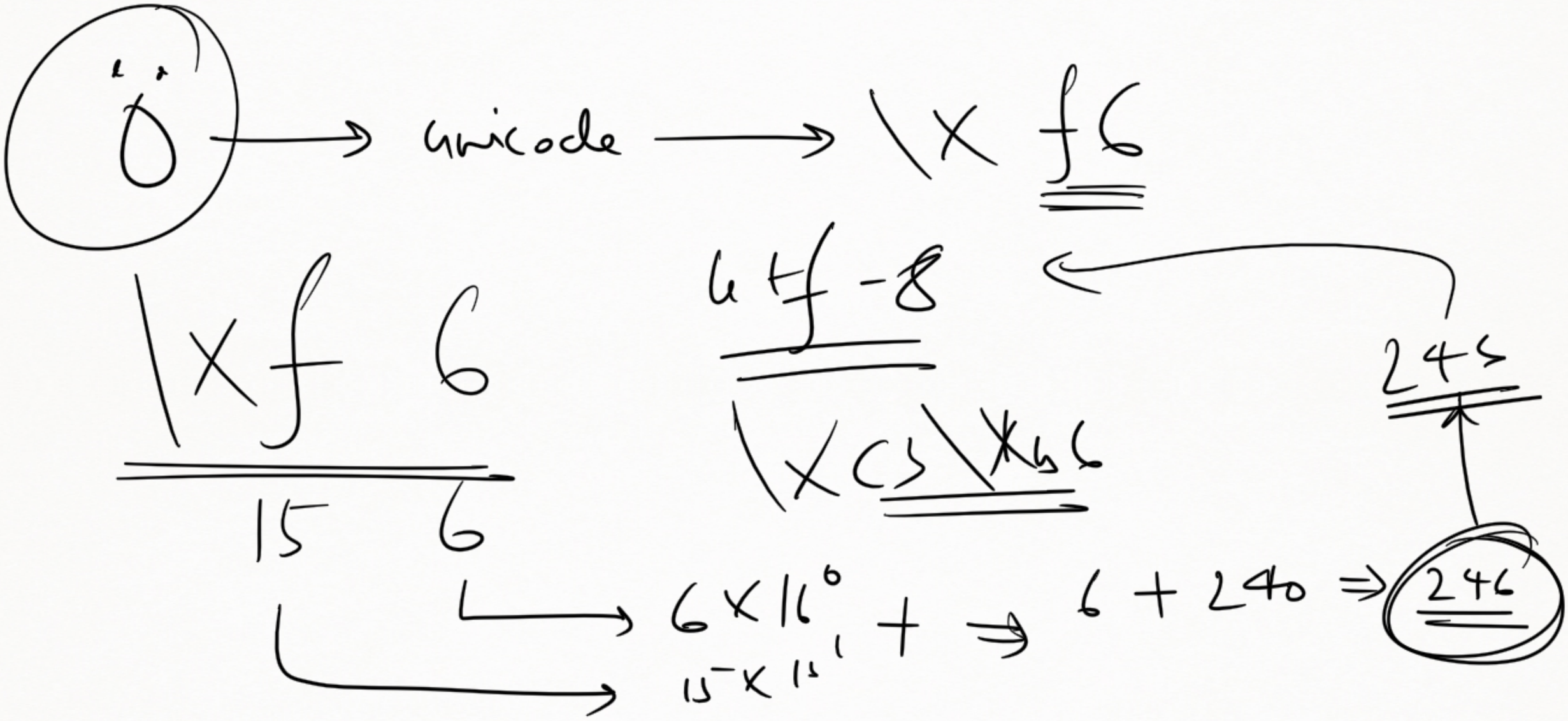
Encode

utf-8

→ unicode → Number


Ö Ó Я ЗЧ
1 1 1
Hexadecimal (Base-16)
\X 3B7D9C
Base 16


0 1 2 3
python
4


ö
unicode
\xf6
utf-8
\xf6
\xc3\xb6
15 6
245
246
6x16^0 +
15x16^1
6 + 240

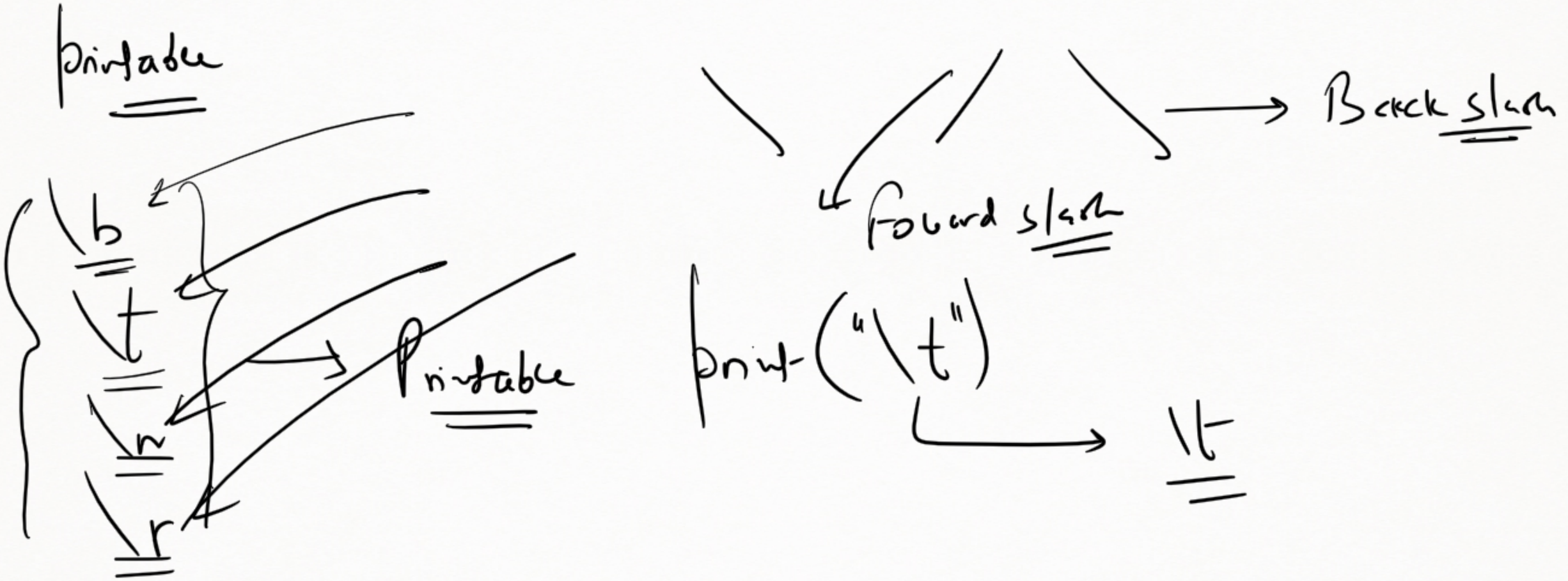
S = "This is going to be amazing"

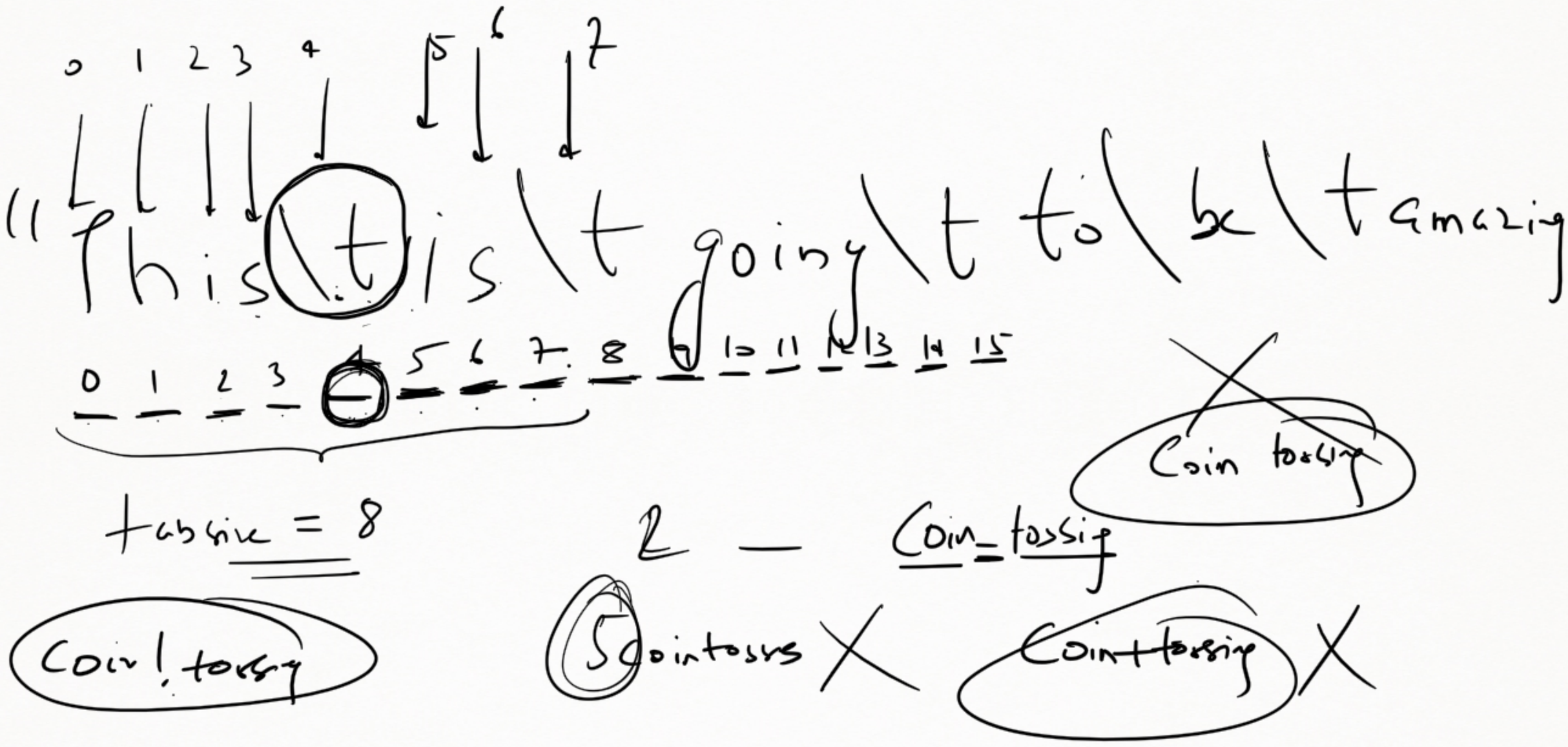
Here are some things inside the string which are not

printable

print(S) ⟶ printed as it is

There are some special type of characters which are not




0 1 2 3 4 5 6 7
"This is going to be amazing
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
tabsize = 8
Coin tossing
Coin_tossing
Coin!tossing
Cointossing
Coin+tossing

(American

Standard

Code for

Information Interchange)

ASCII, UTF-8, Unicode

Unicode UTF-8

What is the problem?

ASCII → It is not an encoding ✓

Unicode → It is not an encoding ✓

→ Mapping

Encoding = ?

Code → Binary (bits)

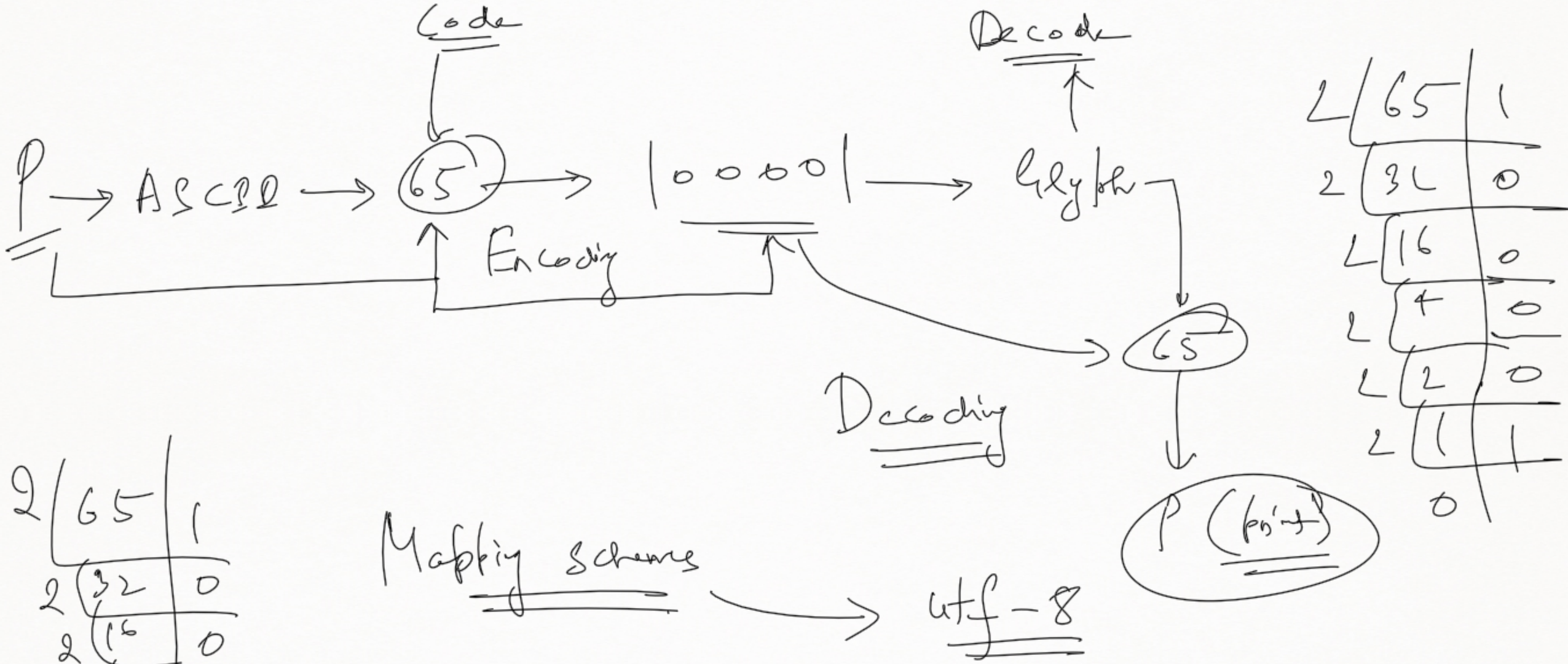
Encoding

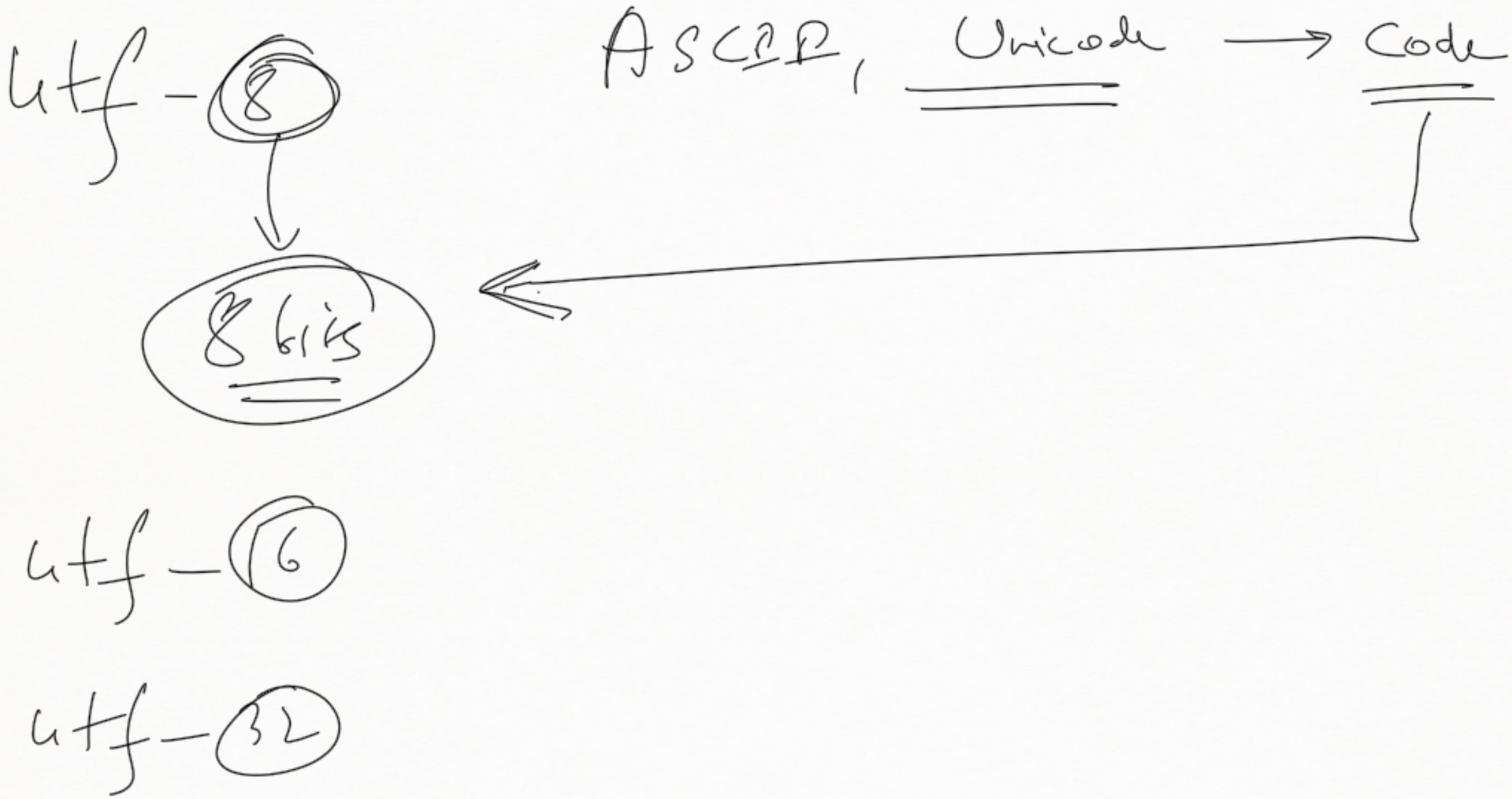
Character → Code

'A' → 65

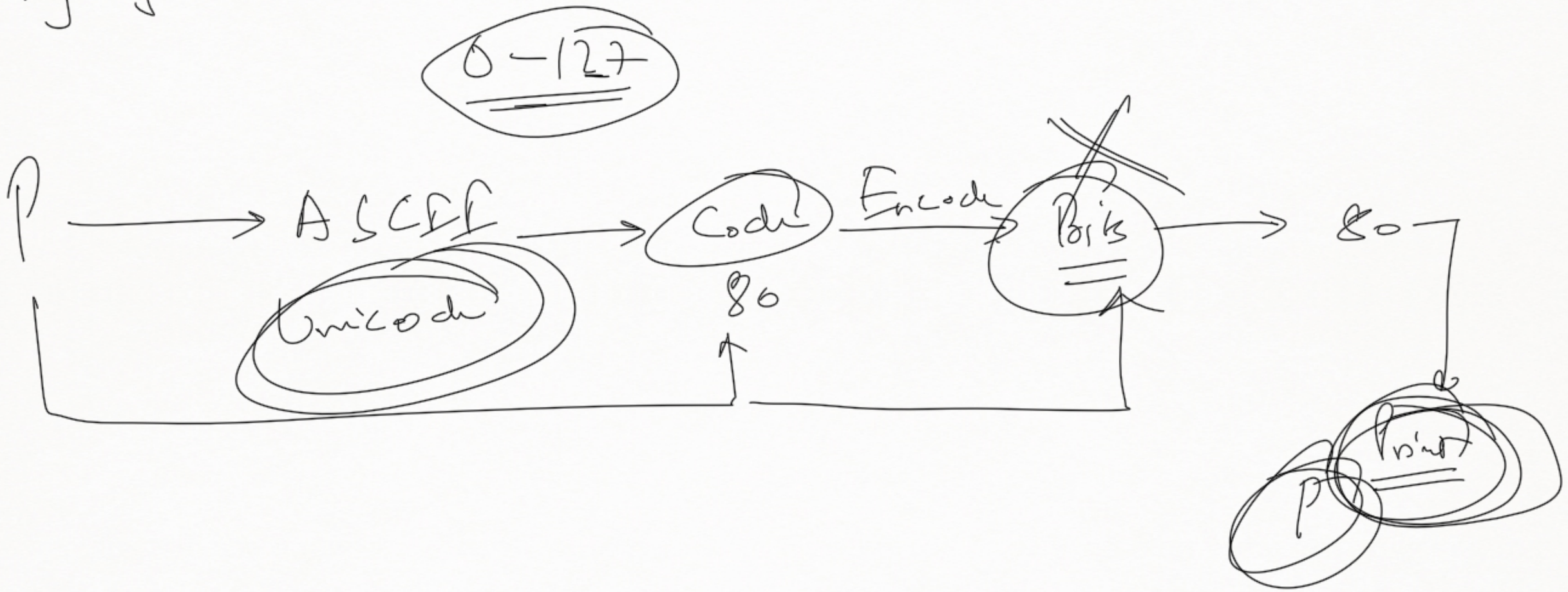
'a' → 97

Bits (Computer can understand)


Code
Decode
P → ASCII → 65 → 100001 → Glyph
Encoding
65
Decoding
P (point)
Mapping scheme
utf-8


utf-8
8 bits
ASCII, Unicode → Code
utf-16
utf-32

If you have a character whose code is between 0 – 127



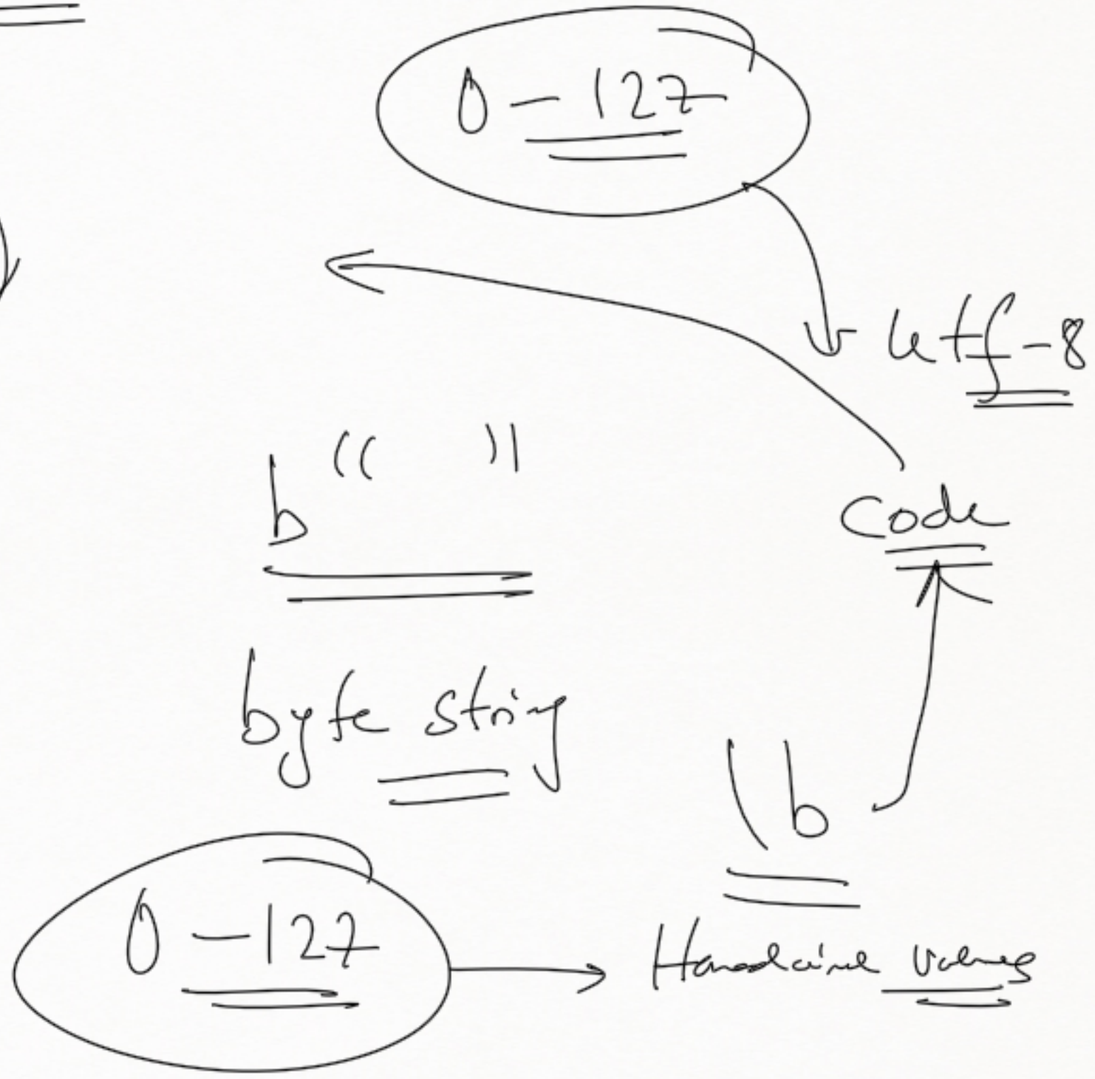
"Python"

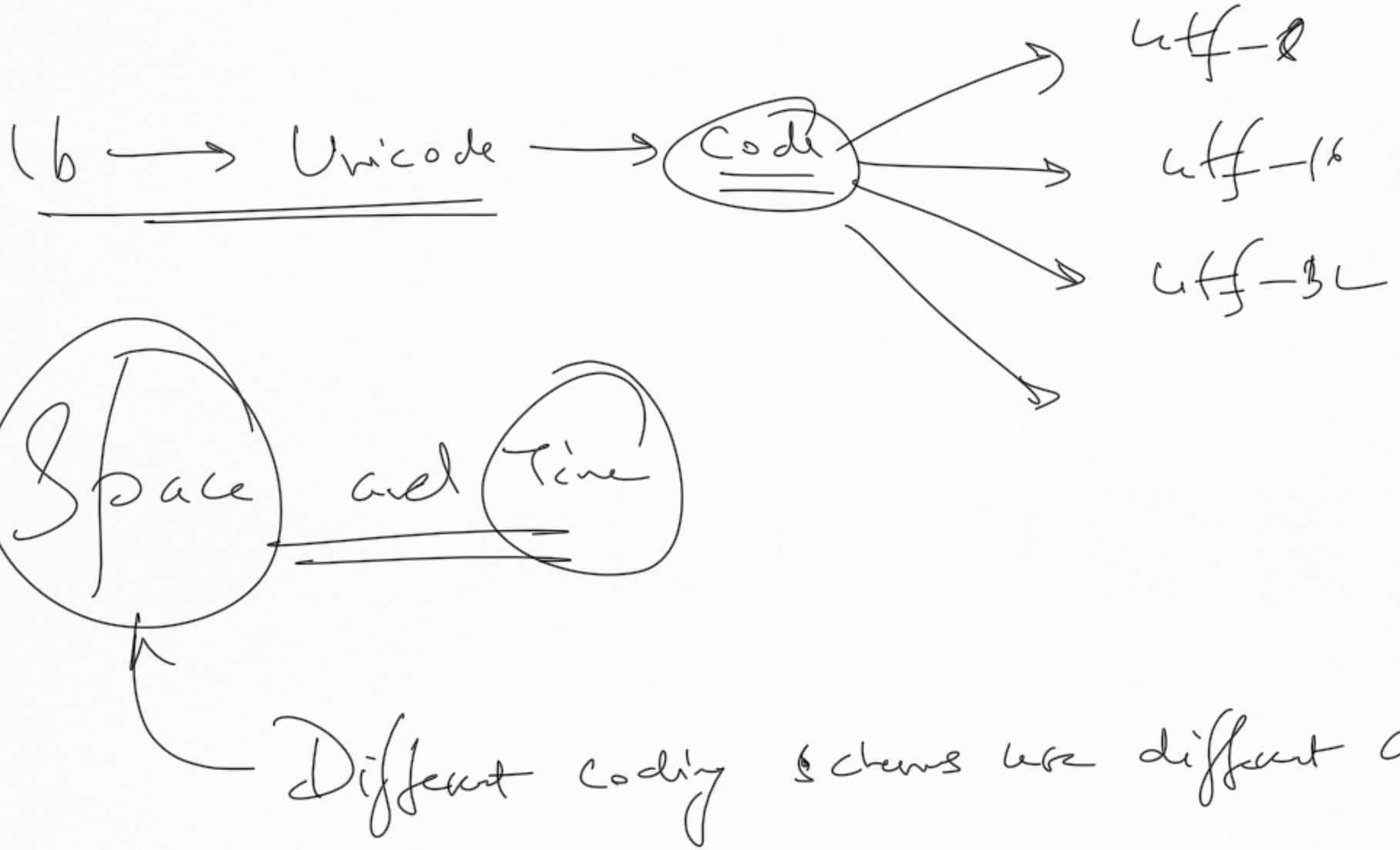
1 Byte = 8 bits

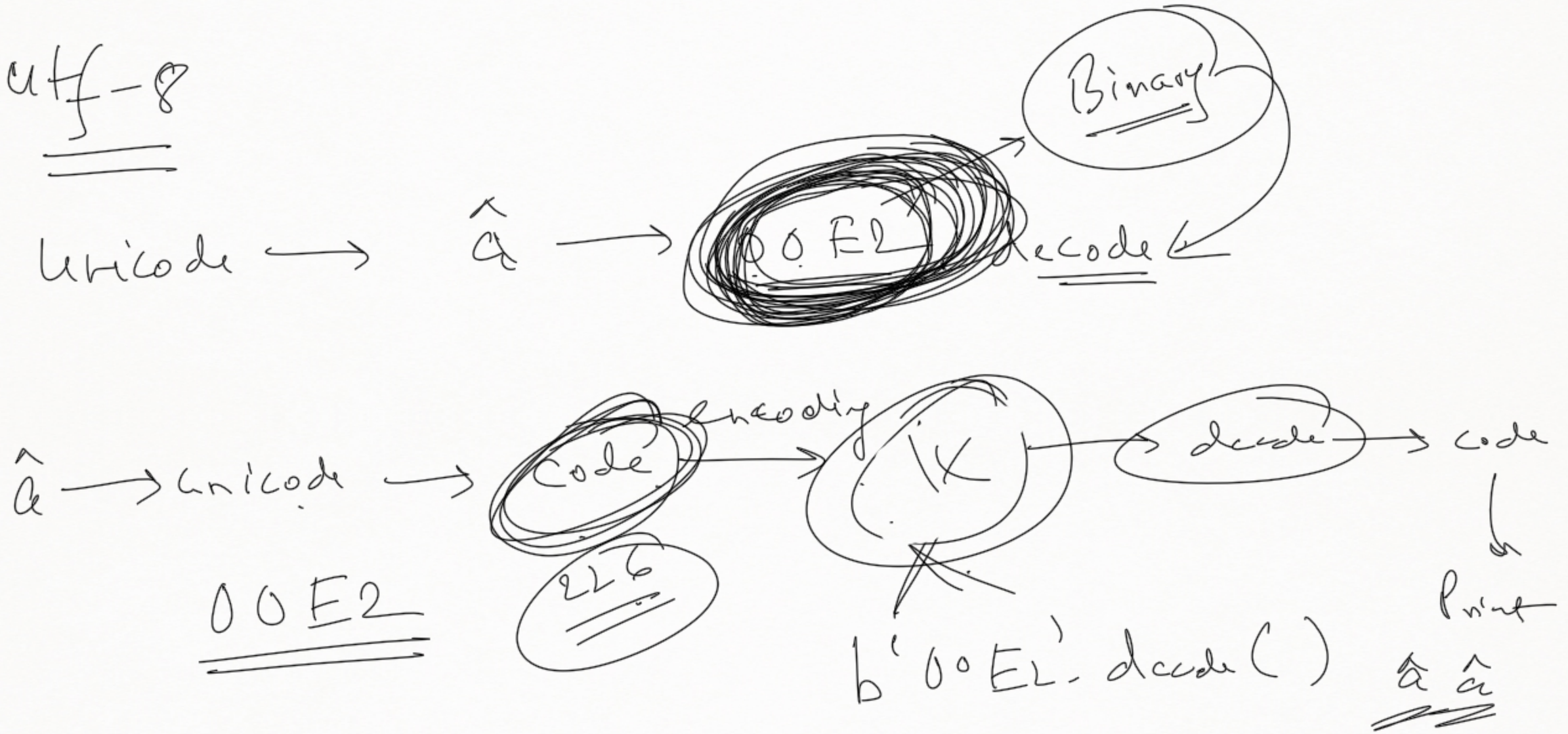
b "Python"

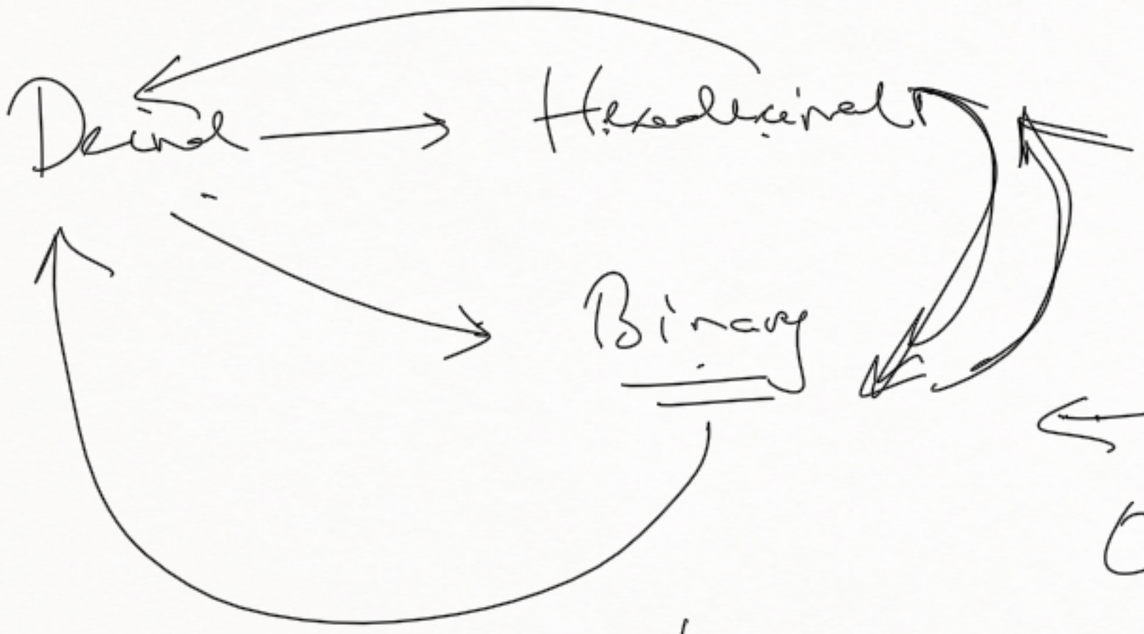
S = "Python"

S.encode()




16 → Unicode → Code
utf-8
utf-16
utf-32
Space and Time
Different coding schemes use different amount of memory


utf-8
Unicode
â
00E2
Binary
recode
â
Unicode
Code
encoding
IX
decode
code
226
00E2
b'00E2'.decode()
Print
â â



16 |
2 |

0

00E2

$2 \times 16^0 = 2 = 2$

$E \times 16^1 = 14 \times 16 = 224$

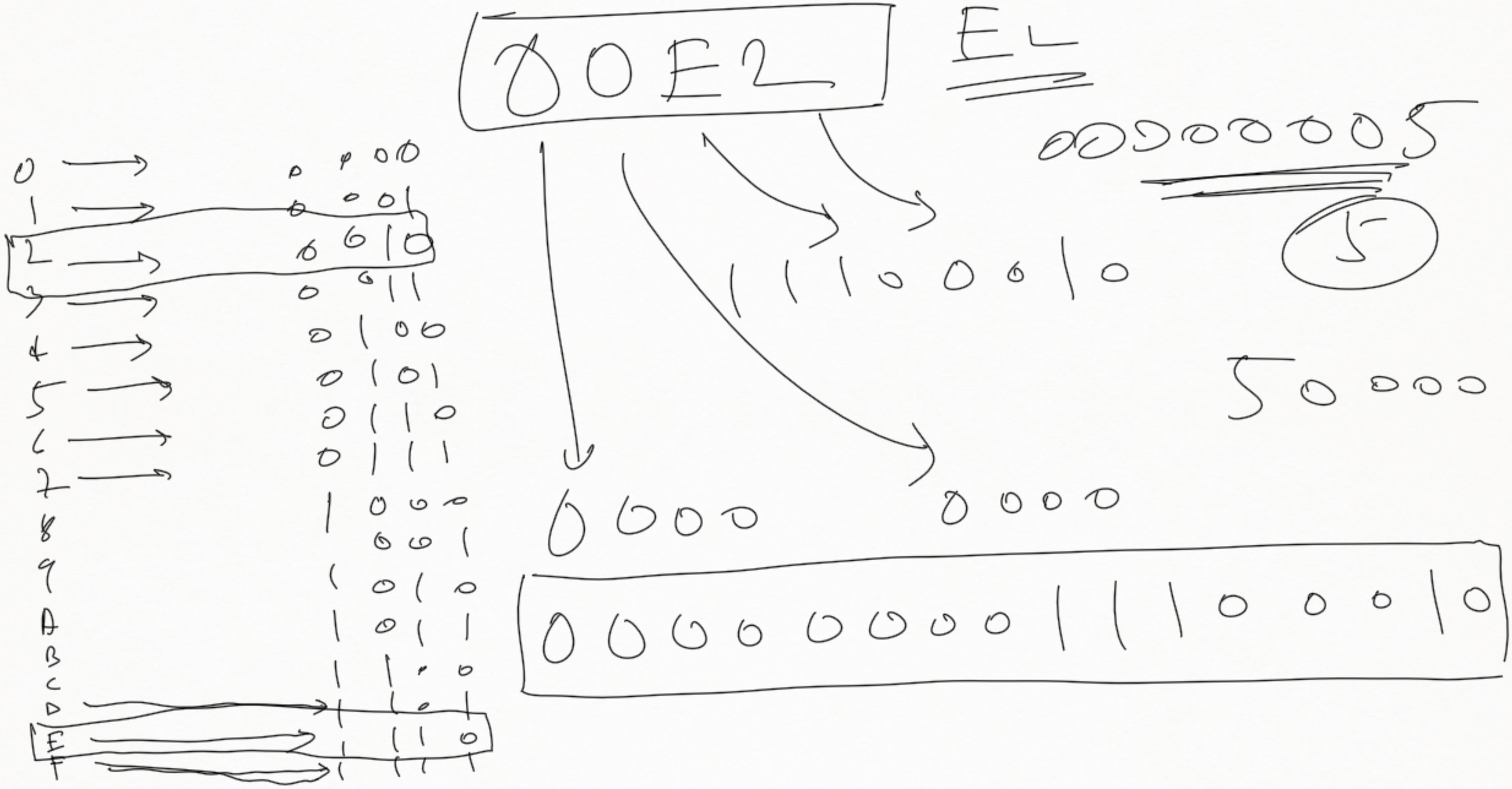
$0 \times 16^2 = 0 = 0$

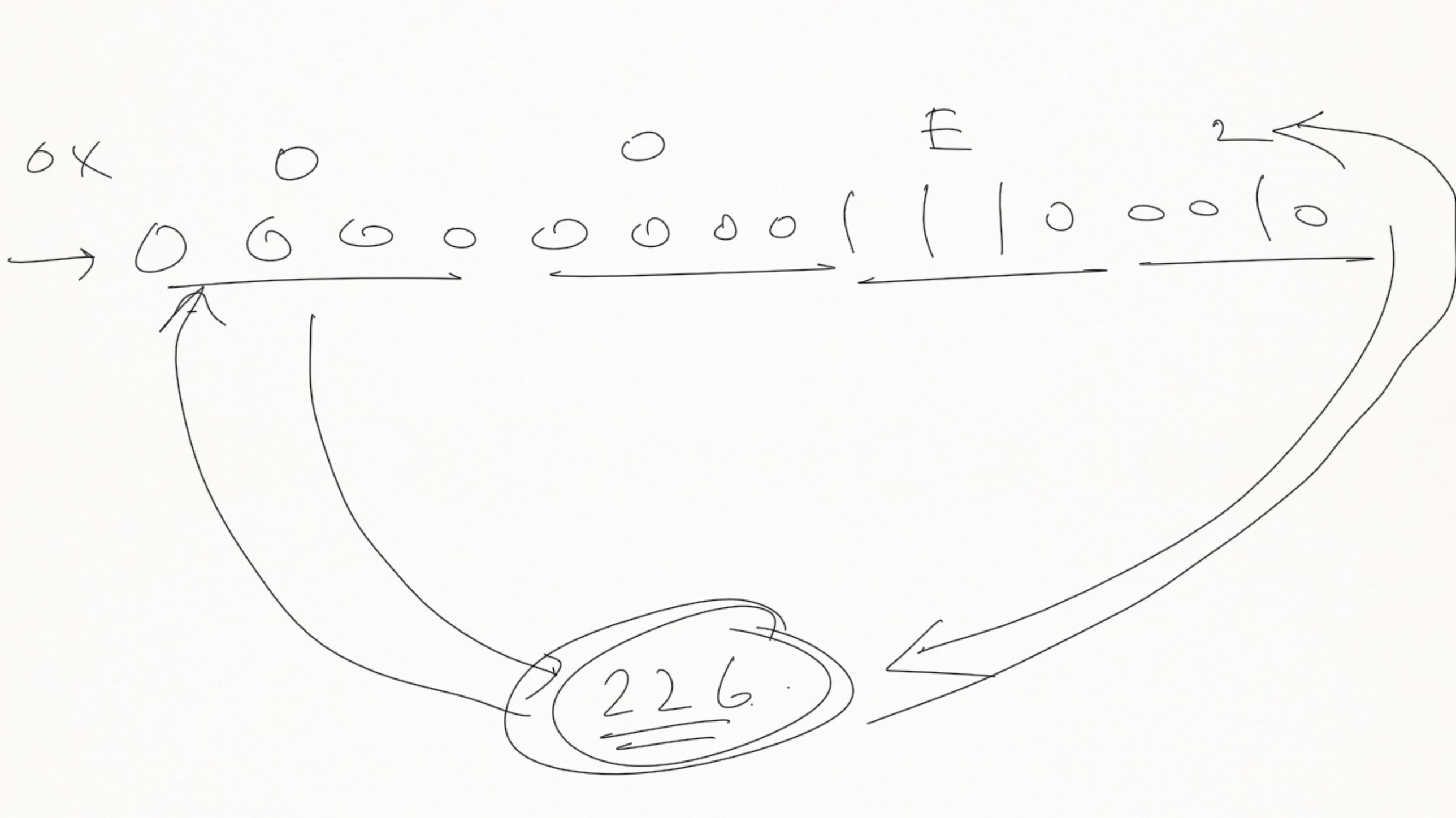
$0 \times 16^3 = 0 = 0$

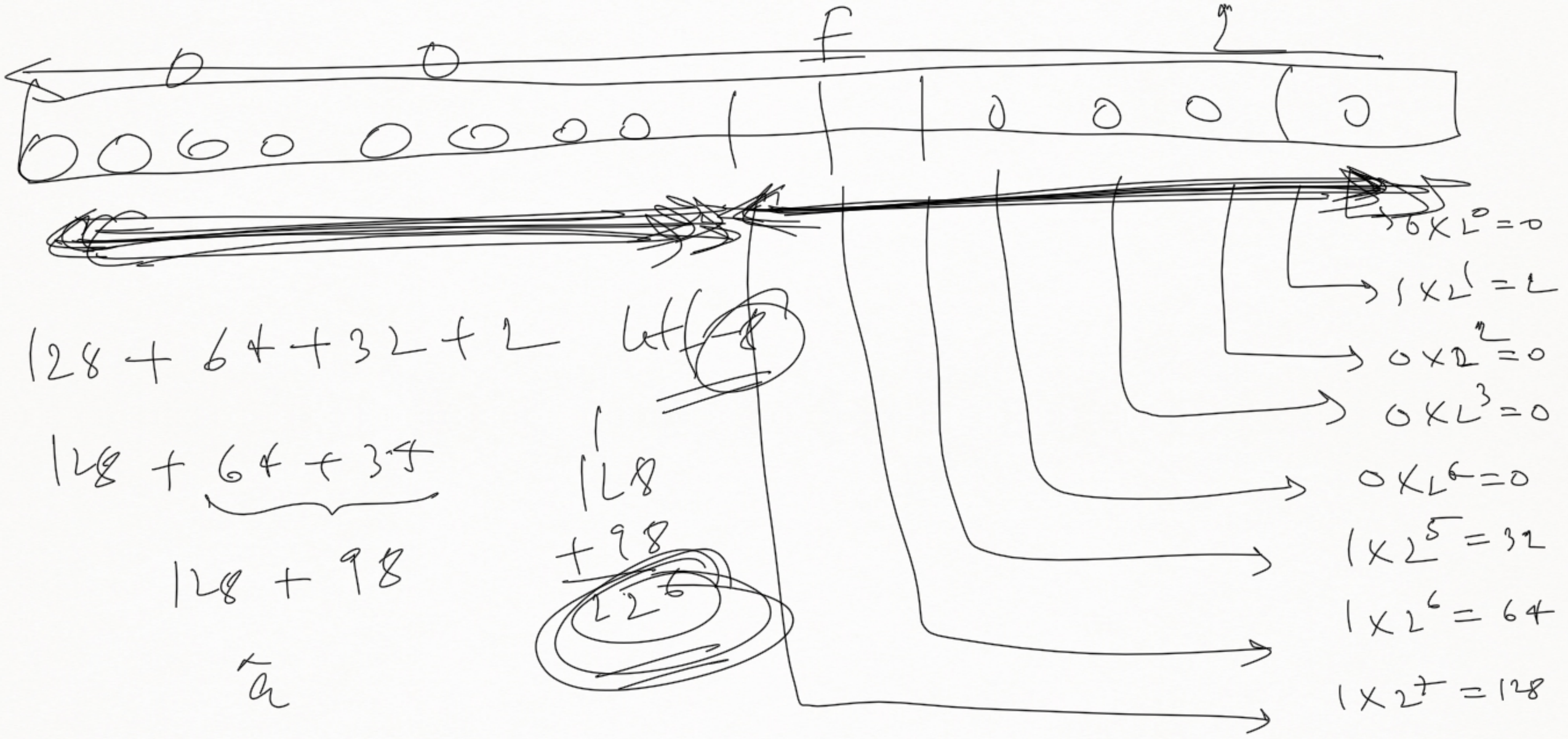
226

E2

16 | 226	2	→ 2
16 | 14	14	→ E
0		


0
1
2
3
4
5
6
7
8
9
A
B
C
D
E
F
00E2
EL
00000005
5
50000


0x
0
0
E
2
226



128 + 64 + 32 + 2

128 + 64 + 34

128 + 98

128
+ 98
226

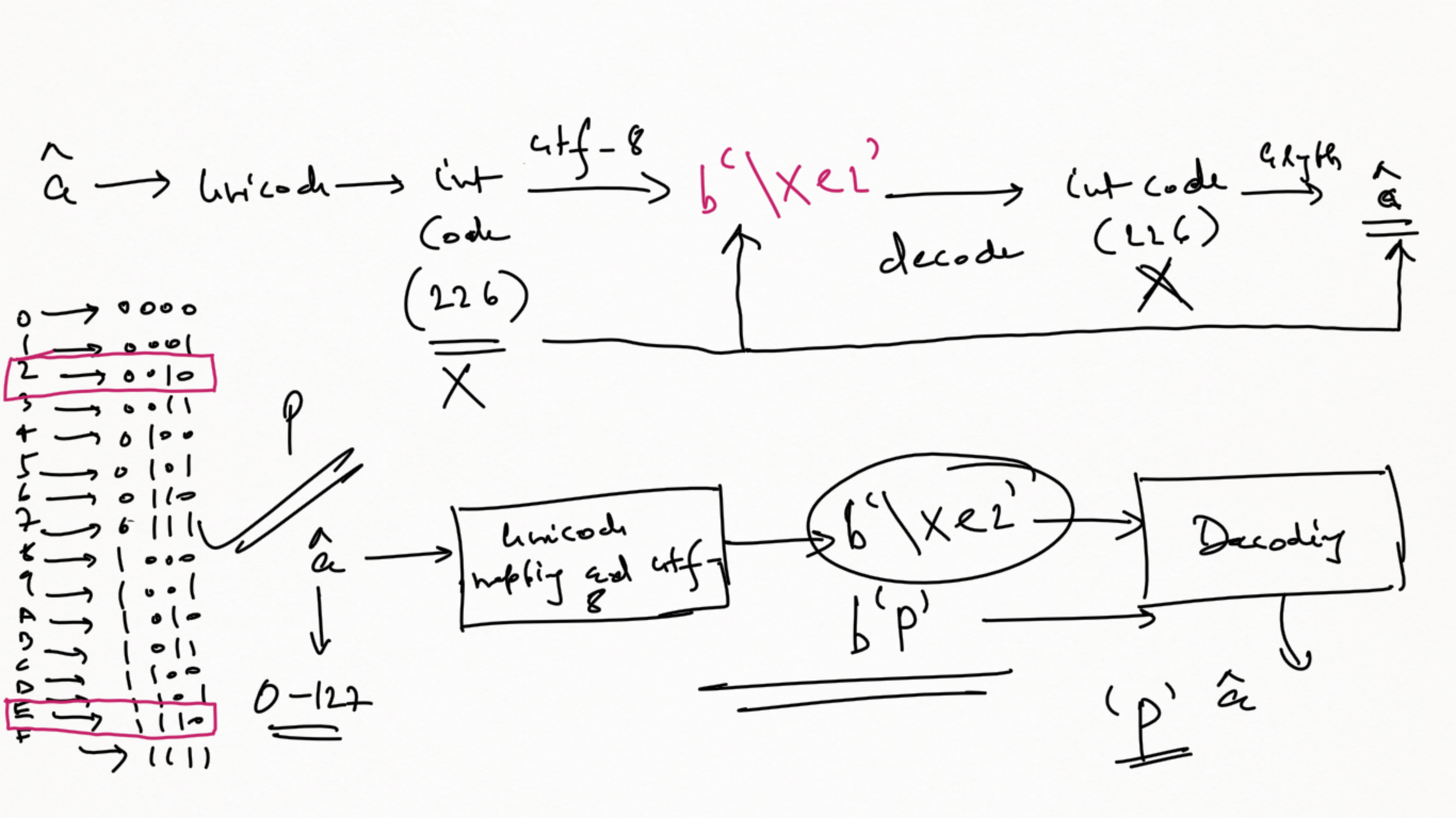
ASCII
Unicode
} → Encoding scheme

A → (Mapping scheme) → Codes (Integers)

A → Unicode → Code → Glyph → Print

A → Unicode → Code → Encode → byte string → Decode → Code → Glyph → Print

utf-8
utf-16, utf-32


â
unicode
int
Code
(226)
X
utf-8
b'\xe2'
decode
int code
(226)
X
Glyph
â
0 → 0000
1 → 0001
2 → 0010
3 → 0011
4 → 0100
5 → 0101
6 → 0110
7 → 0111
8 → 1000
9 → 1001
A → 1010
B → 1011
C → 1100
D → 1101
E → 1110
F → 1111
p
â
0-127
unicode
mapping and utf-8
b'\xe2'
b'p'
Decoding
p'
â

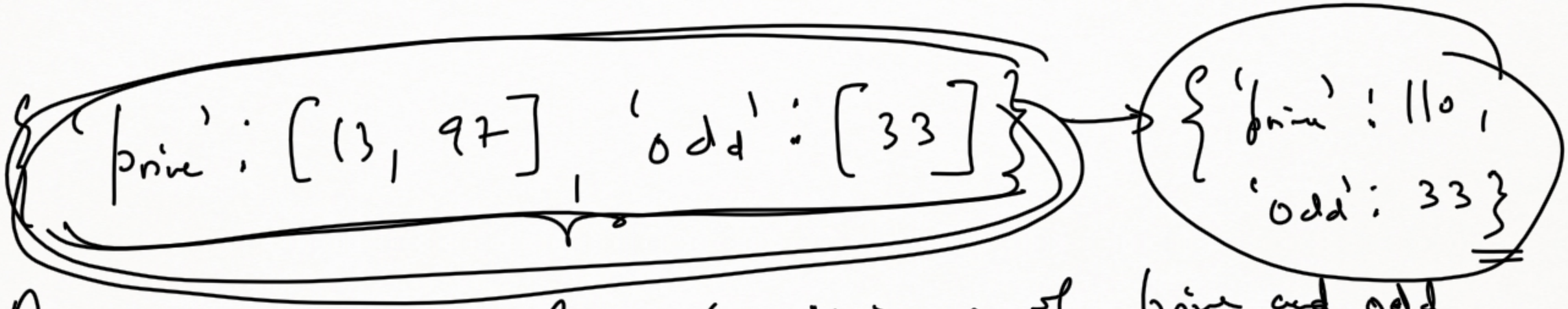
$$y = f(x) = \sin x$$

$$y = f(x) = x^3 + 3x^2 + 9x + 7$$

$$y = f(x) = \begin{cases} \cos x, & \text{if } x^3 + 3x^2 + 9x + 7 > 0 \text{ for } x > 0 \\ \tan x & \text{if } x^3 + 3x^2 + 9x + 7 < 0 \text{ for } \underline{\underline{x < 0}} \end{cases}$$

Numbers = [13, 97, 54, 33, 52]

500 lines



Assignment → Create the dictionary of prime and odd numbers and convert this dictionary into the dictionary of sum of prime and odd numbers

list → append, sort, count, pop, remove, extend, copy

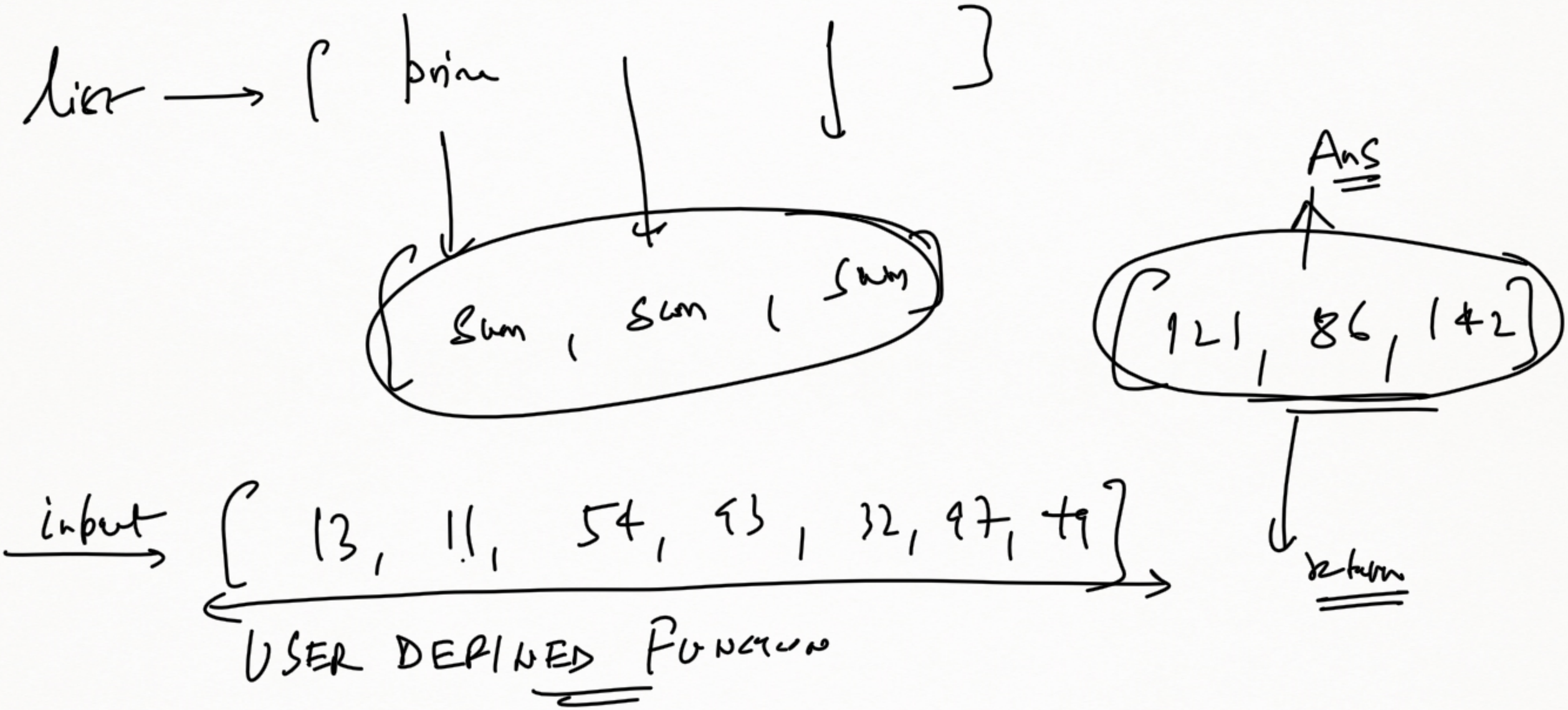
dict →

tuple →

set →

str →

PREDEFINED FUNCTIONS


list → [prime]
sum , sum , sum
Ans
121 , 86 , 142
input
[13 , 11 , 54 , 93 , 32 , 97 , 49]
return
USER DEFINED FUNCTION