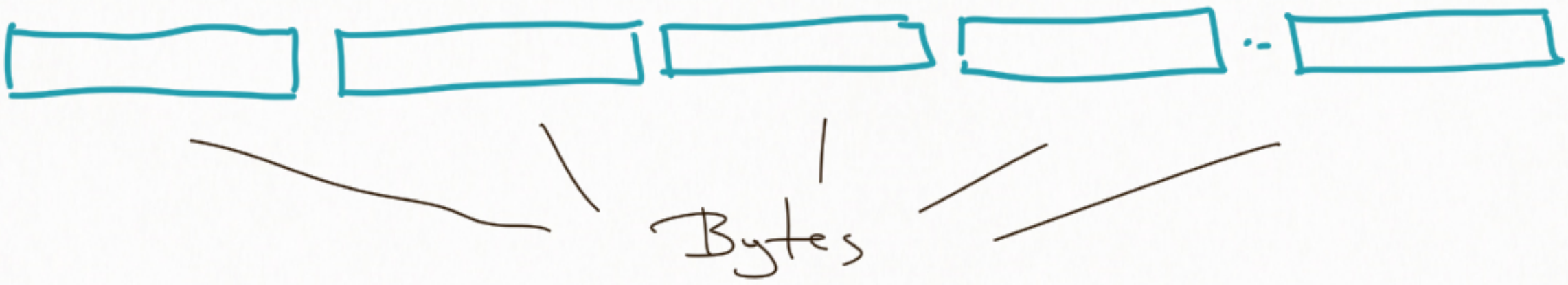
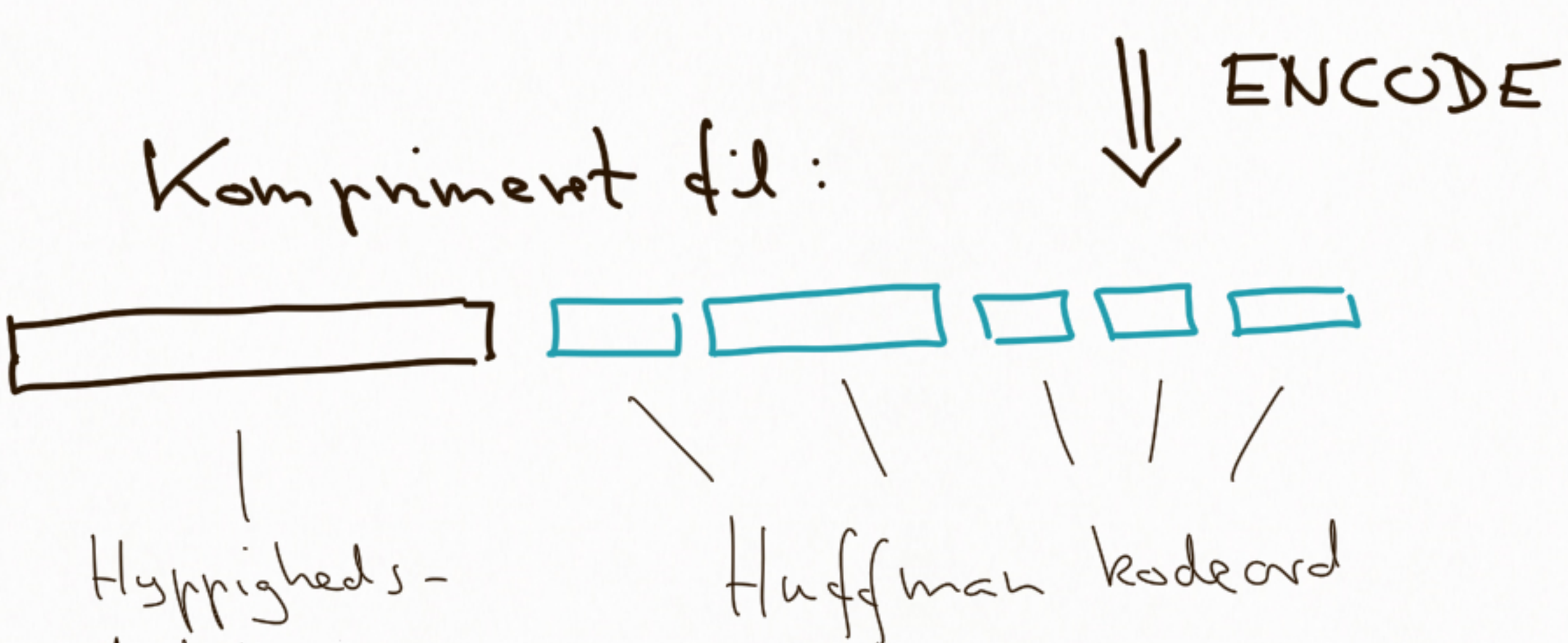
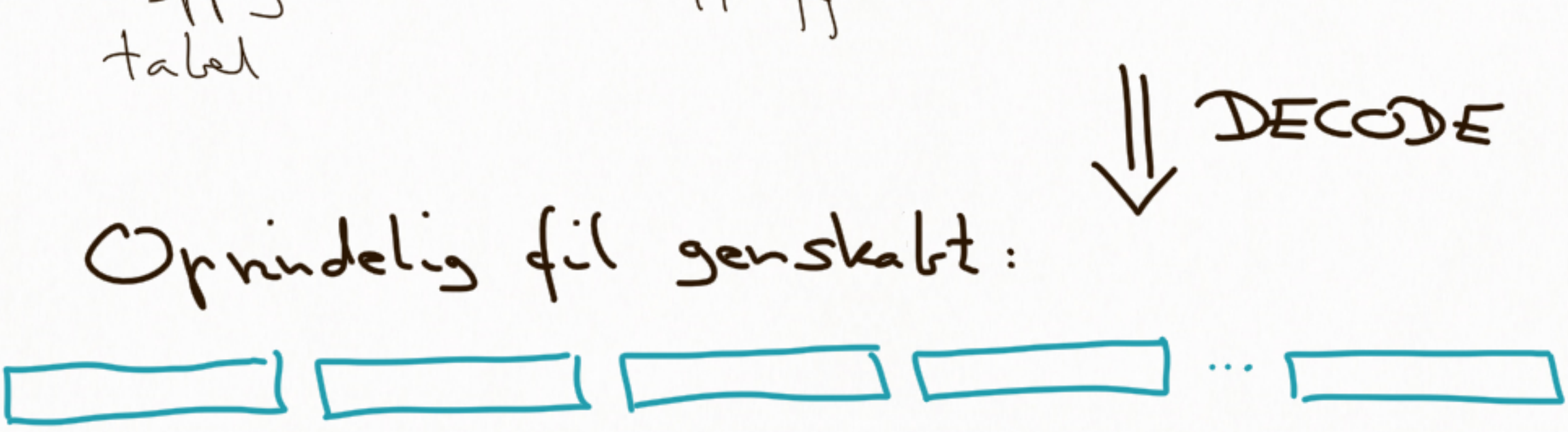
Projekt del III som tegneserie

IDE :

Opprindelig fil :

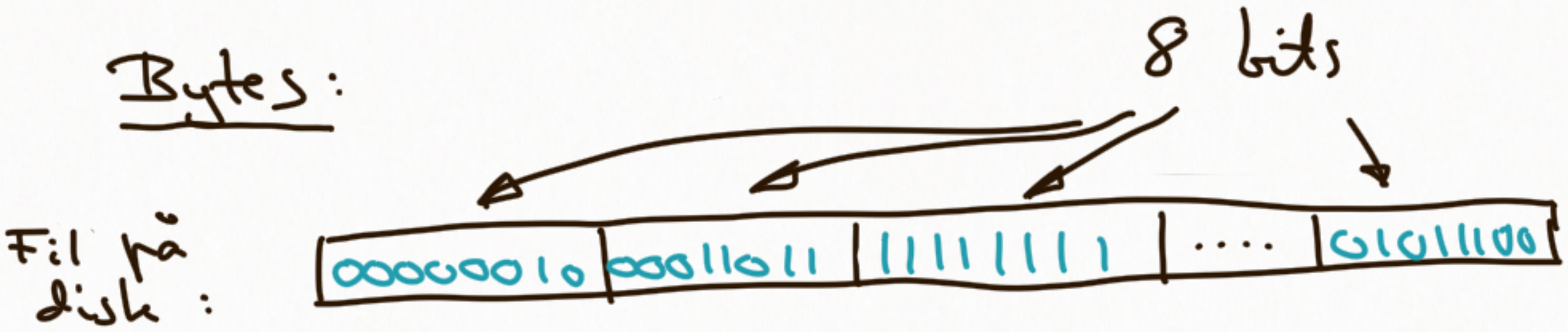






ENCODE

Bytes:



read() [Java]
read(1) [Python]

I program: 2 27 255 92

ints (heltal)

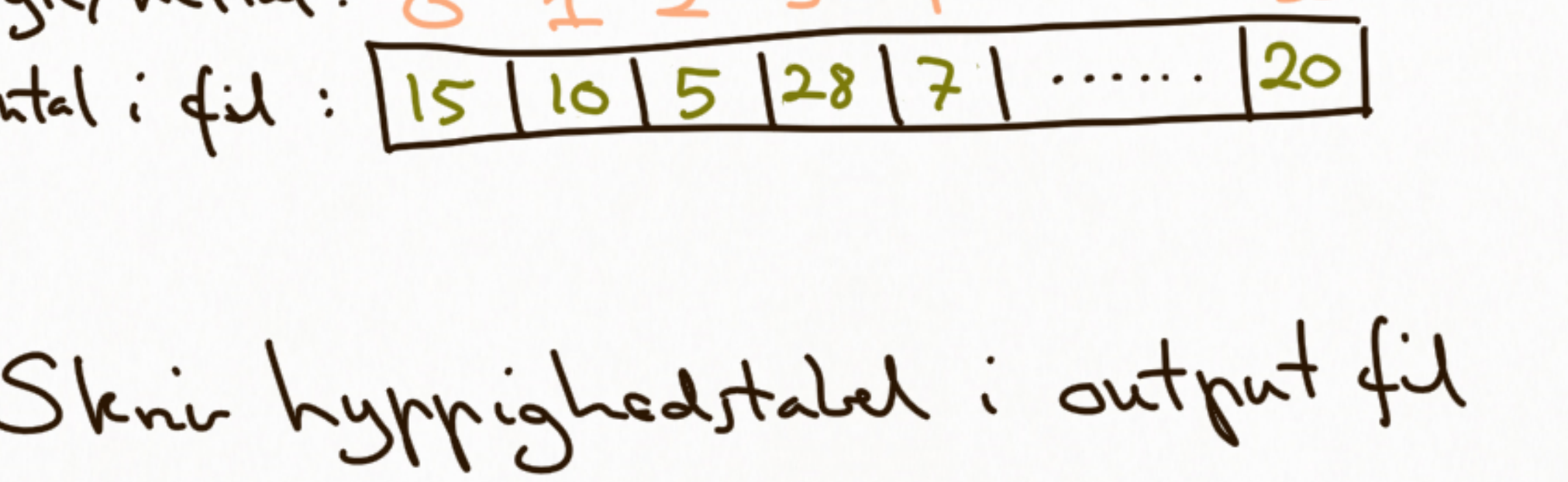
Tæl antal bytes i fil af hver type

Hyppighedstabel:

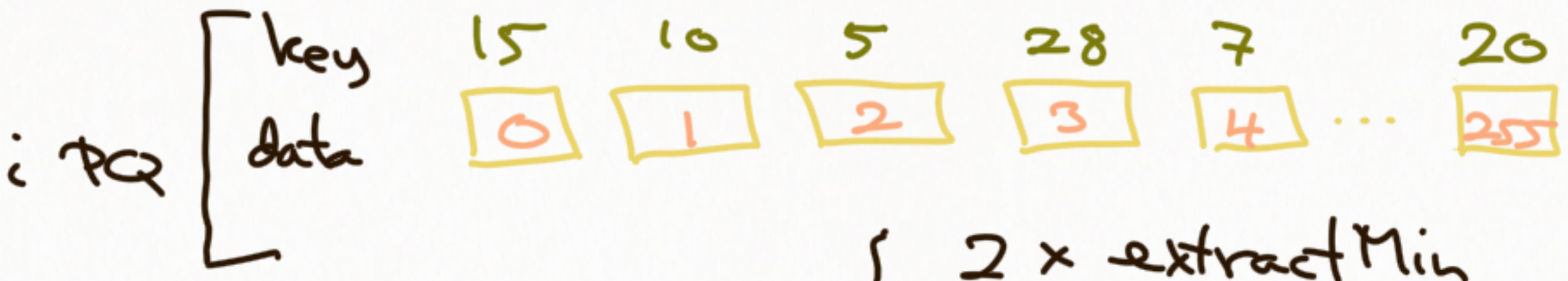
byte/heltal:	0	1	2	3	4	...	255
antal i fil:	15	10	5	28	7	.....	20

Skriv hyppighedstabel i output fil

writeInt(i) [Java]
writeint32bits(i) [Python]



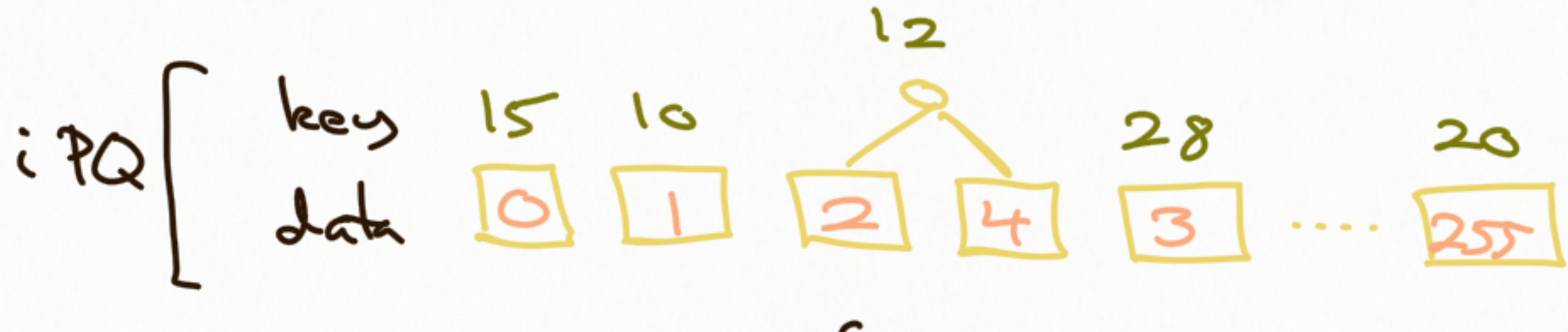
Kør Huffmans algoritme på hyppighedstabel:



2 × extractMin

Tilføj rod

1 × insert



Gentag til ét træ tilbage

⋮

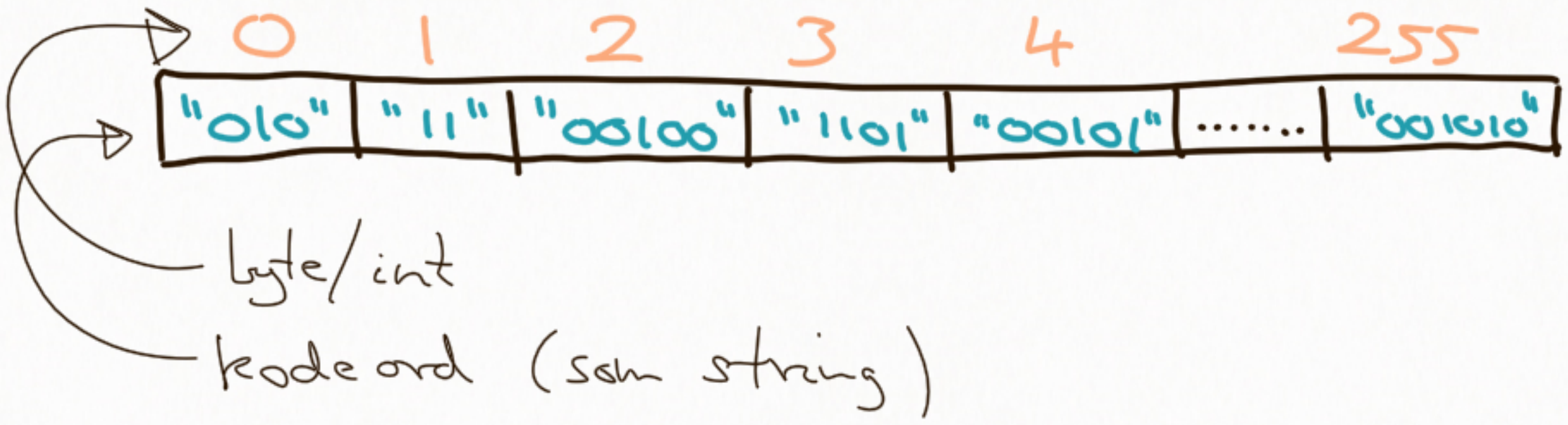
⟹ Huffmantræ

Kodeord for 4 er 00101

Gennemløb Huffmantræ, vedligehold sti som string, udskriv når blade mødes [jvf. opg. 1 fra Uge 17]

⇓

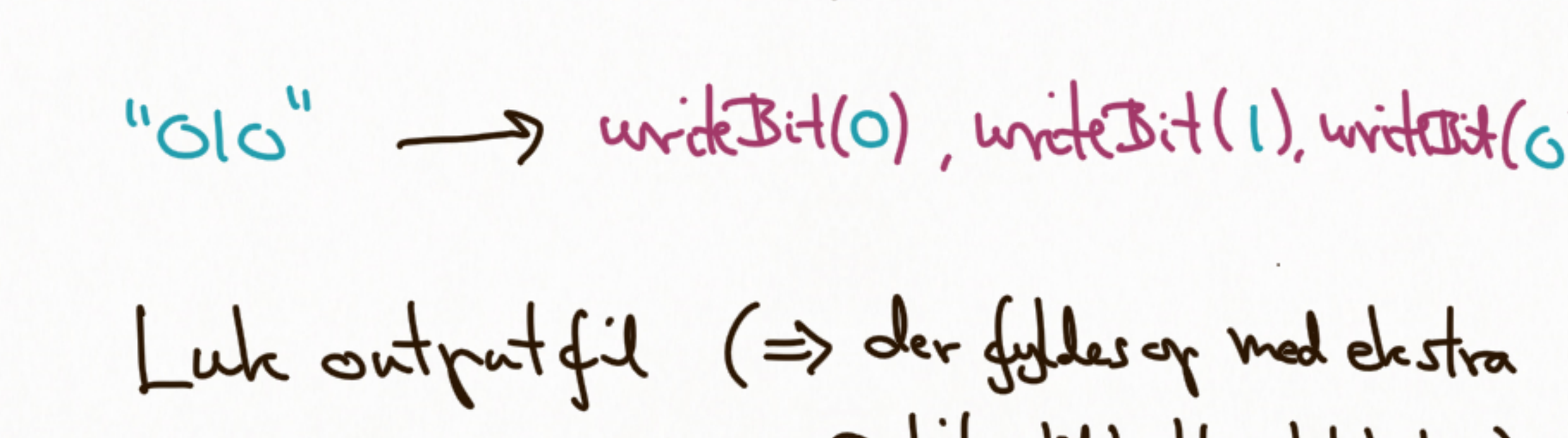
Tabel over kodeord:



Gennemløb fil igen, for hvert byte slå kodeord op i tabel, løb kodeord igennem tegn for tegn, skriv bits i output fil:

"010" ⟶ writeBit(0), writeBit(1), writeBit(0)

Luk outputfil (⇒ der fyldes op med ekstra 0-bits til helt antal bytes)



DEKODE

Læs hyppighedstabel (= første del af kodede fil).

readInt() [Java]

read int 32 bits() [Python]

Samme hyppighedstabel som før.

Lav Huffmantræ ud fra den, brug samme program/kode som før.

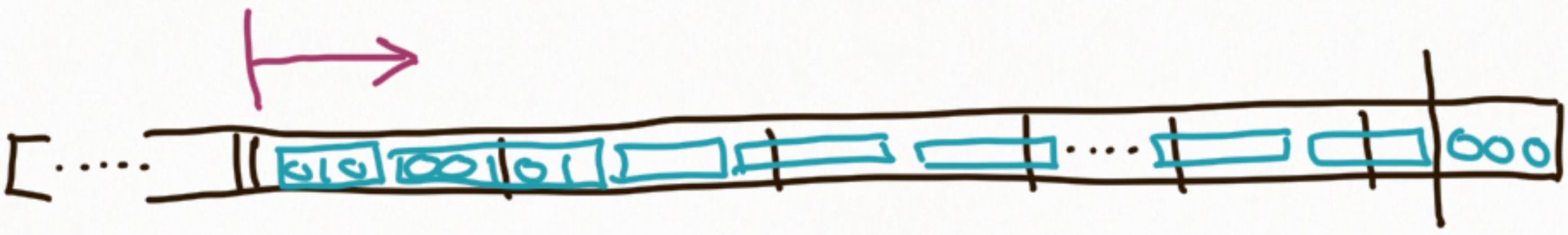
Samme Huffmanstræ

Find også summen af tallene i hyppighedstabellen (= antal bytes i den oprindelige fil).

Læs resten af filen bit for bit

readBit() [Java]

readbit() [Python]

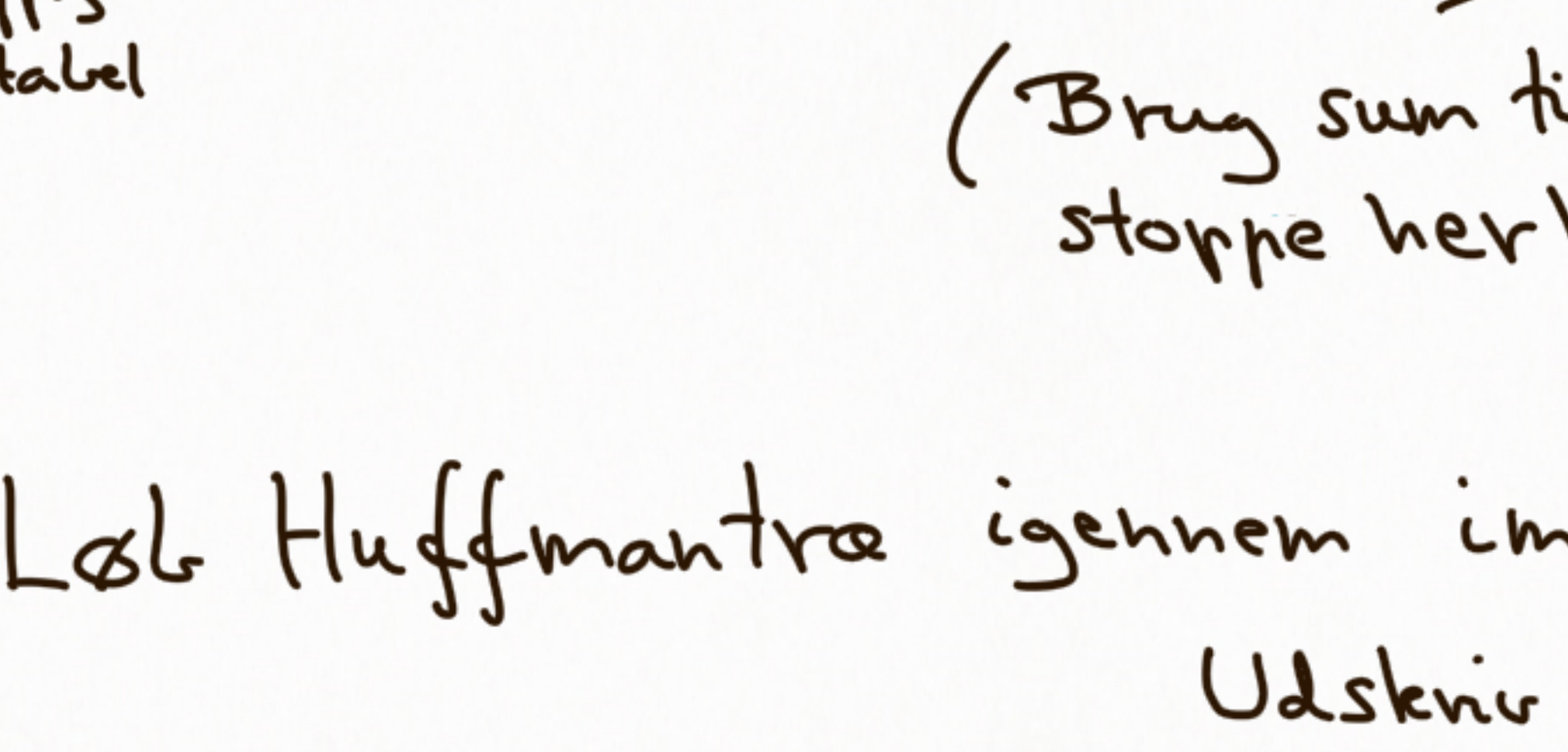


Hyppig-hedstabel

stop

(Brug sum til at stoppe her)

Læs Huffmantræ igennem imens. Udskriv bogstav når blad nås og gå videre fra roden.



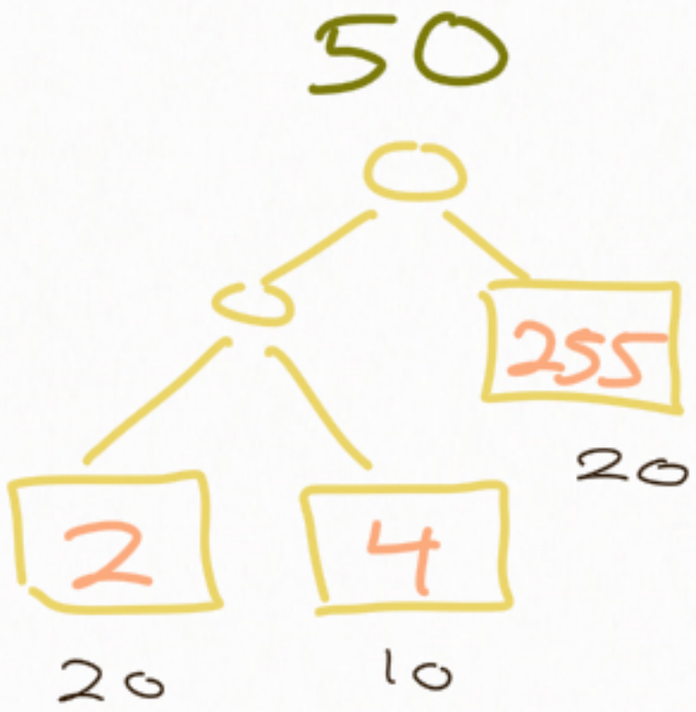
write(4) [Java]

write(bytes([4])) [Python]

⟹ Oprindelige fil genskabt.

(10, [4])

(30, [[2], [4]])



(50, [[[2], [4]], [255]])