

# DE24- Programming 40 Yhp

Lopar

# Uppprepningar & Loopar

- Med upprepning och loopar så syftar vi på en kontrollstruktur som låter oss upprepa ett block av kod så länge ett visst villkor är uppfyllt
- Detta ger oss en möjlighet att automatisera repetitiva uppgifter som skulle leda till att vi måste ha duplicerad kod efter varandra
- Python stödjer 2 typer av loopar, 'for'- loop och 'while' - loop

# for-loop

''' En for-loop består av först: nyckelordet 'for' därefter variabelnamnet. Vanligt kallas variabeln för 'i' men kan i princip vara vad som helst. Variabelnamnet kommer ta emot objektet i sekvensen. Därefter kommer nyckelordet 'in'. Sist kommer den sekvens som ska itireras igenom och avslutas med kolon. All indenterad kod under kommer köras!

Exempel:

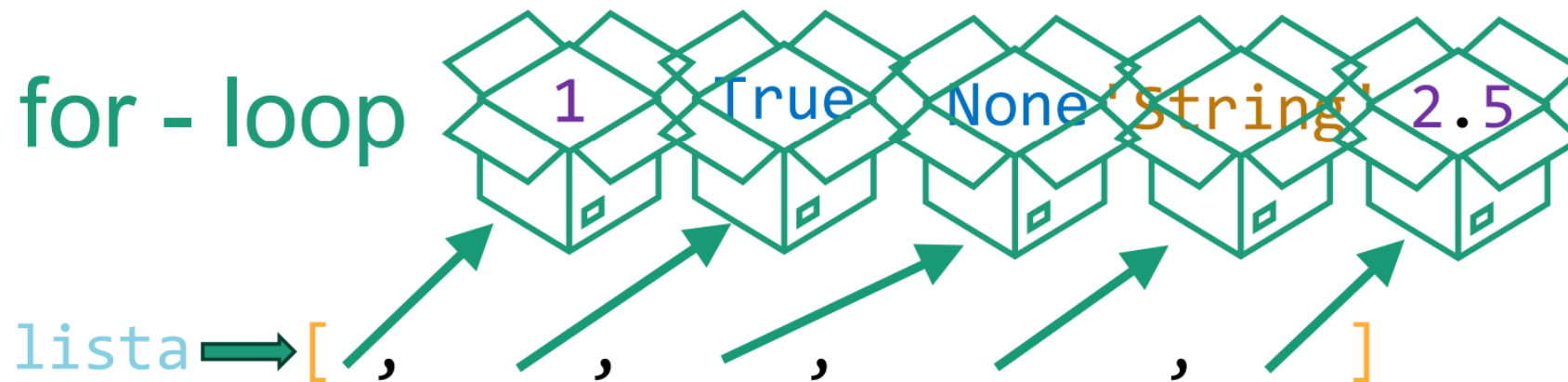
```
'''  
for element in sekvens:  
    print(element)
```

# for-loop

```
lista = [1, True, None, 'String', 2.5]
```

```
for i in lista:  
    print(i)
```

# for-loop



```
for i in lista:  
    print(i)
```

# for-loop

for - loop

lista

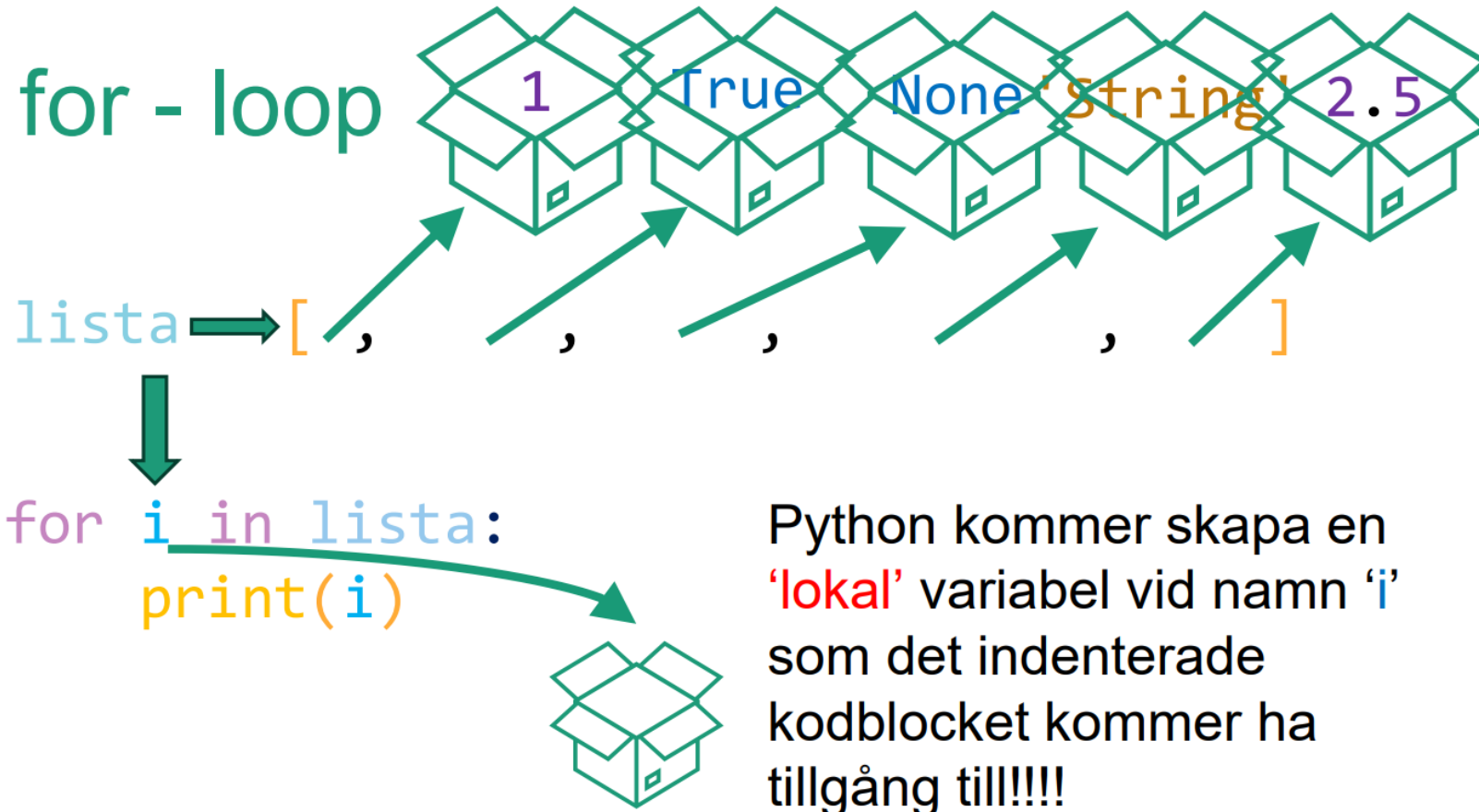
[ , , , , ]



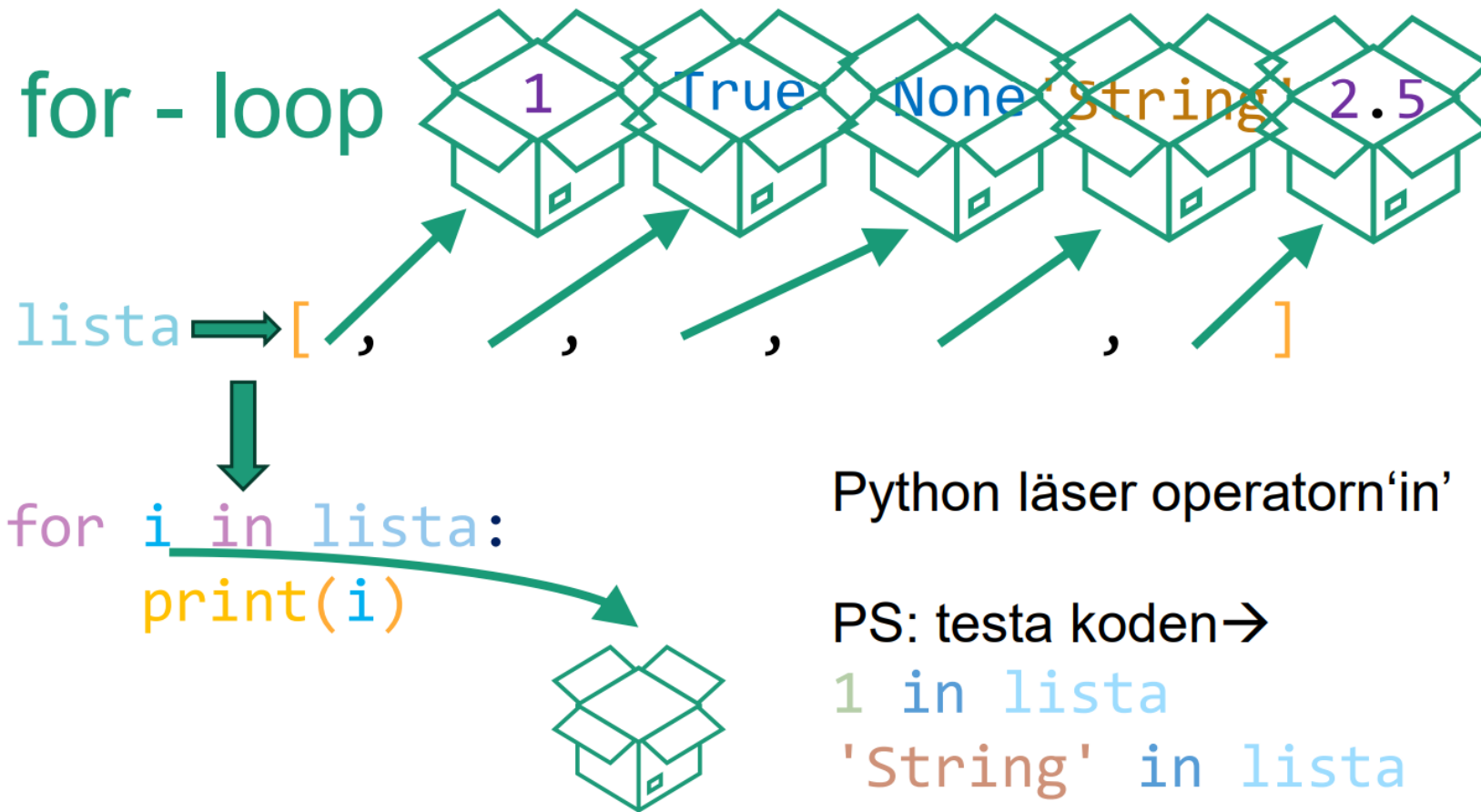
```
for i in lista:  
    print(i)
```



# for-loop

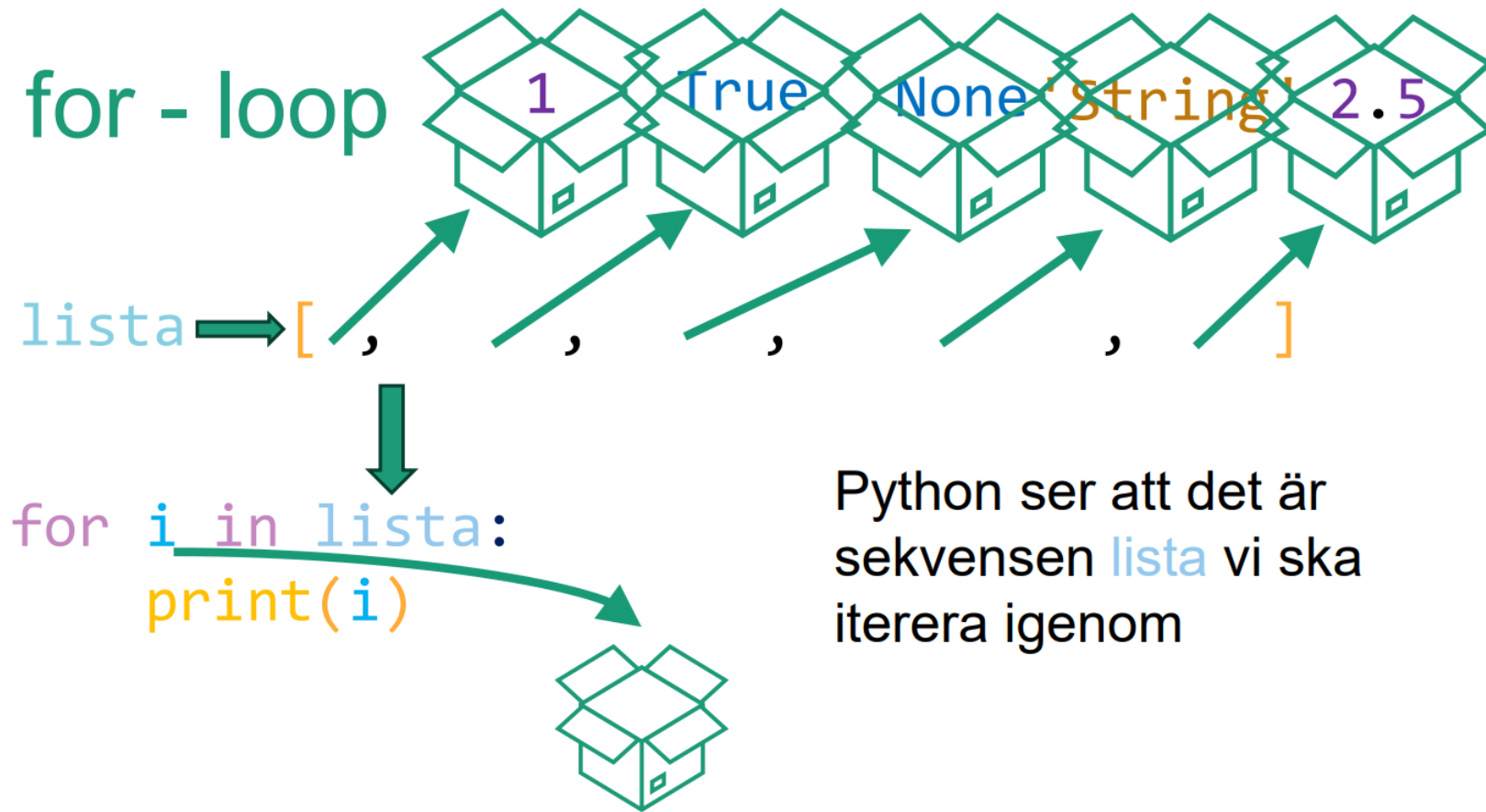


# for-loop

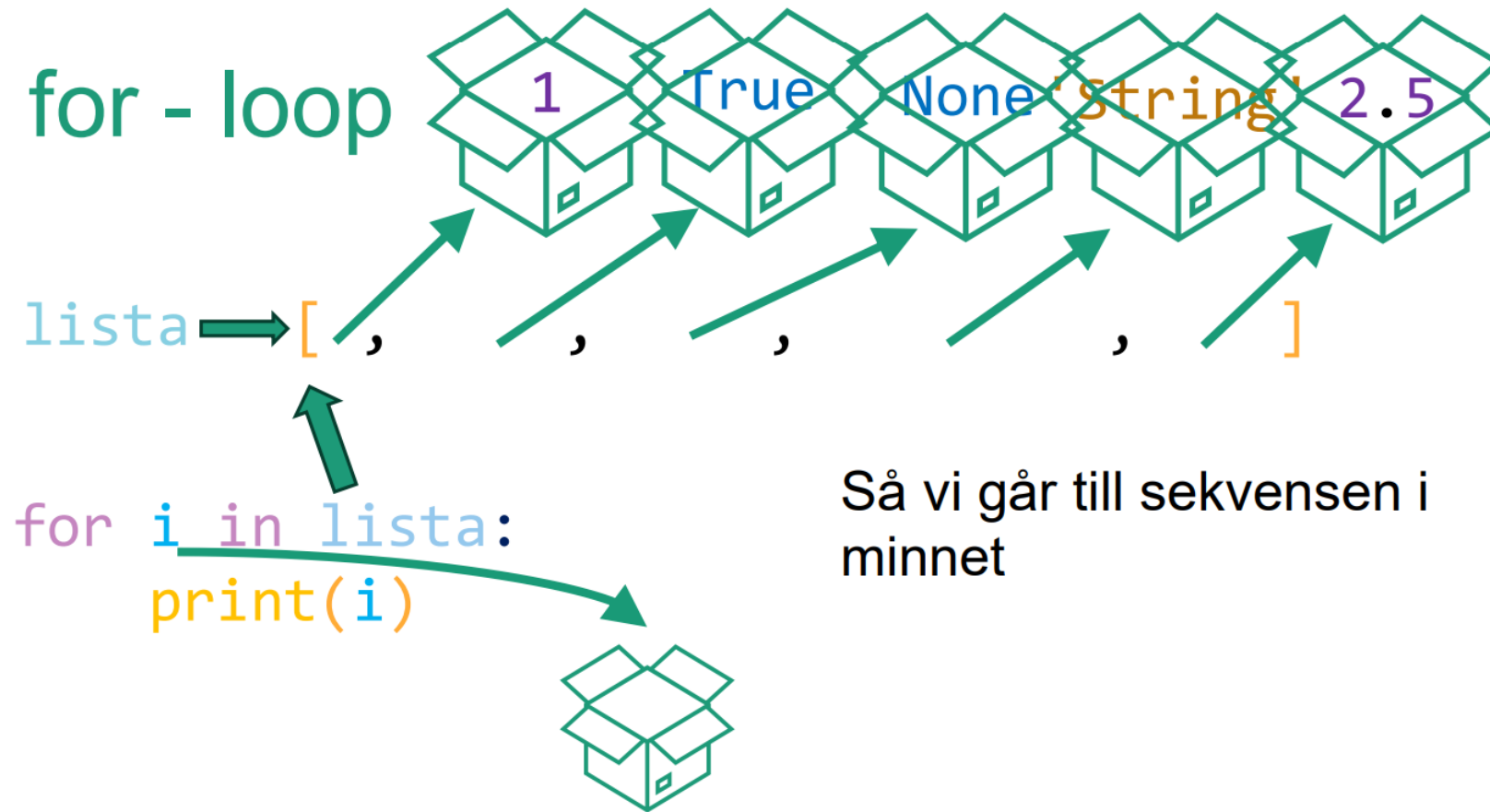




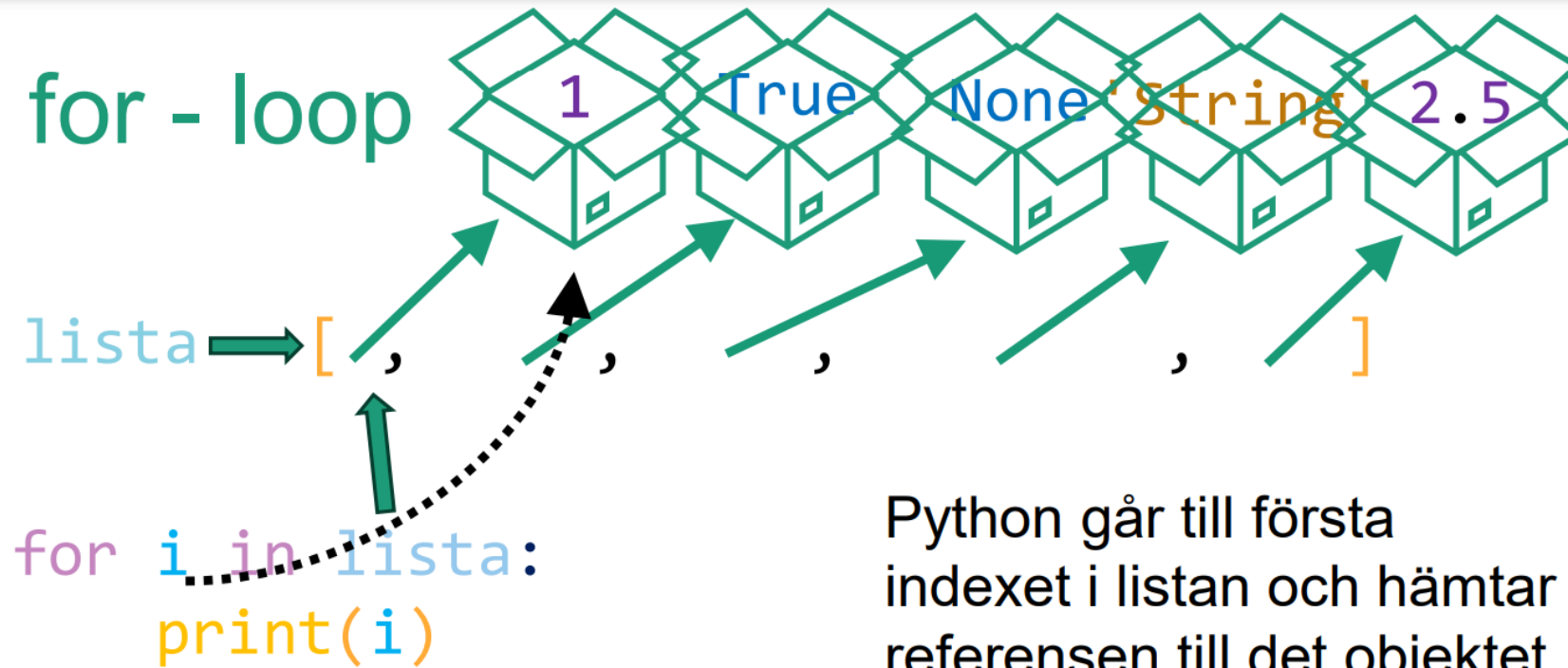
# for-loop



# for-loop

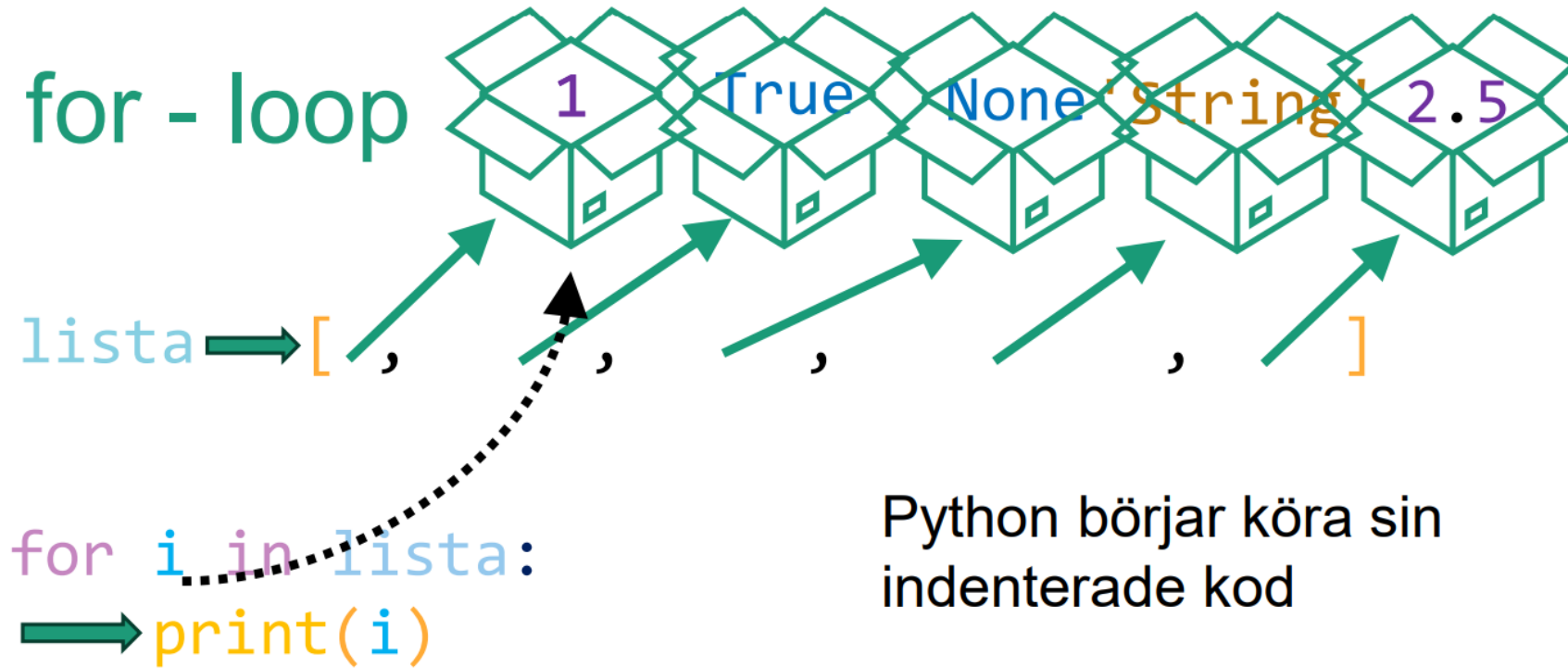


# for-loop

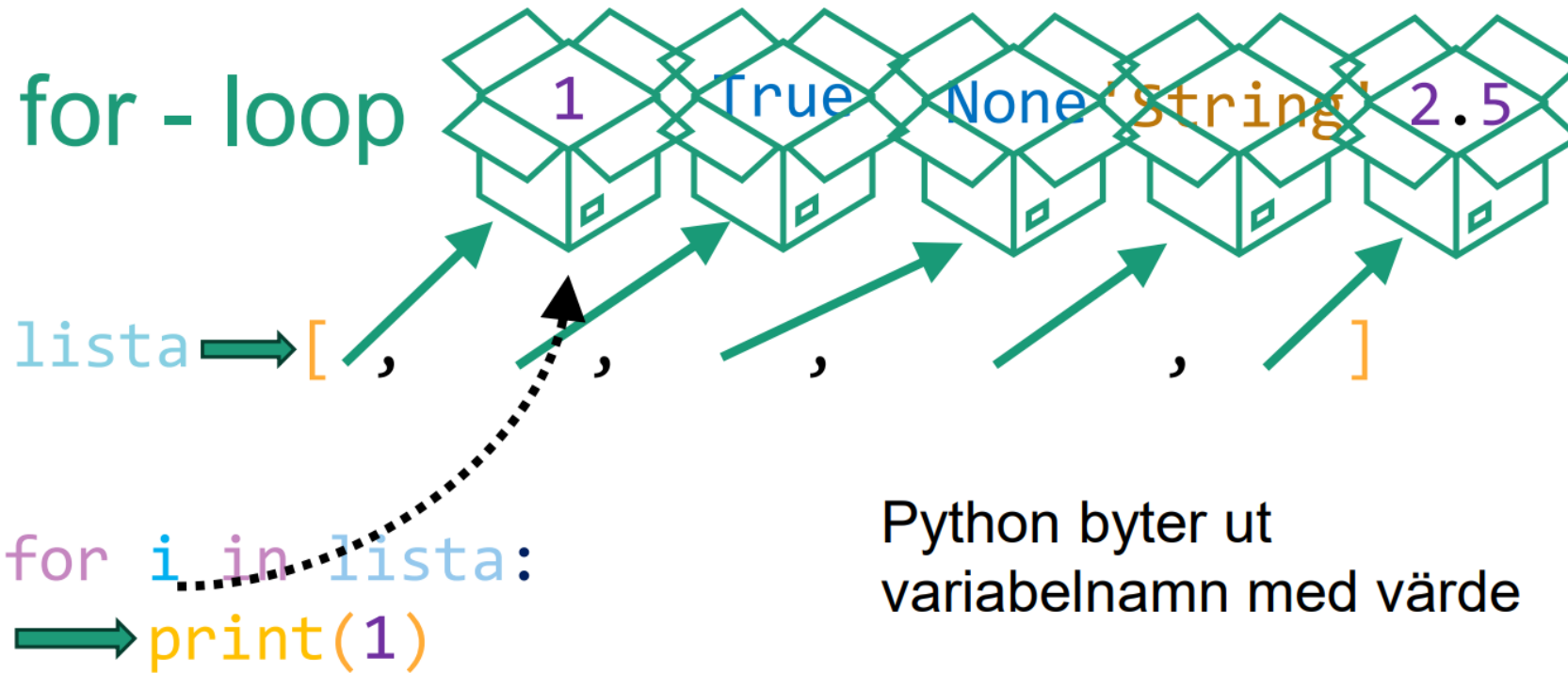


Python går till första indexet i listan och hämtar referensen till det objektet och tilldelar den lokala variabeln 'i' det värdet

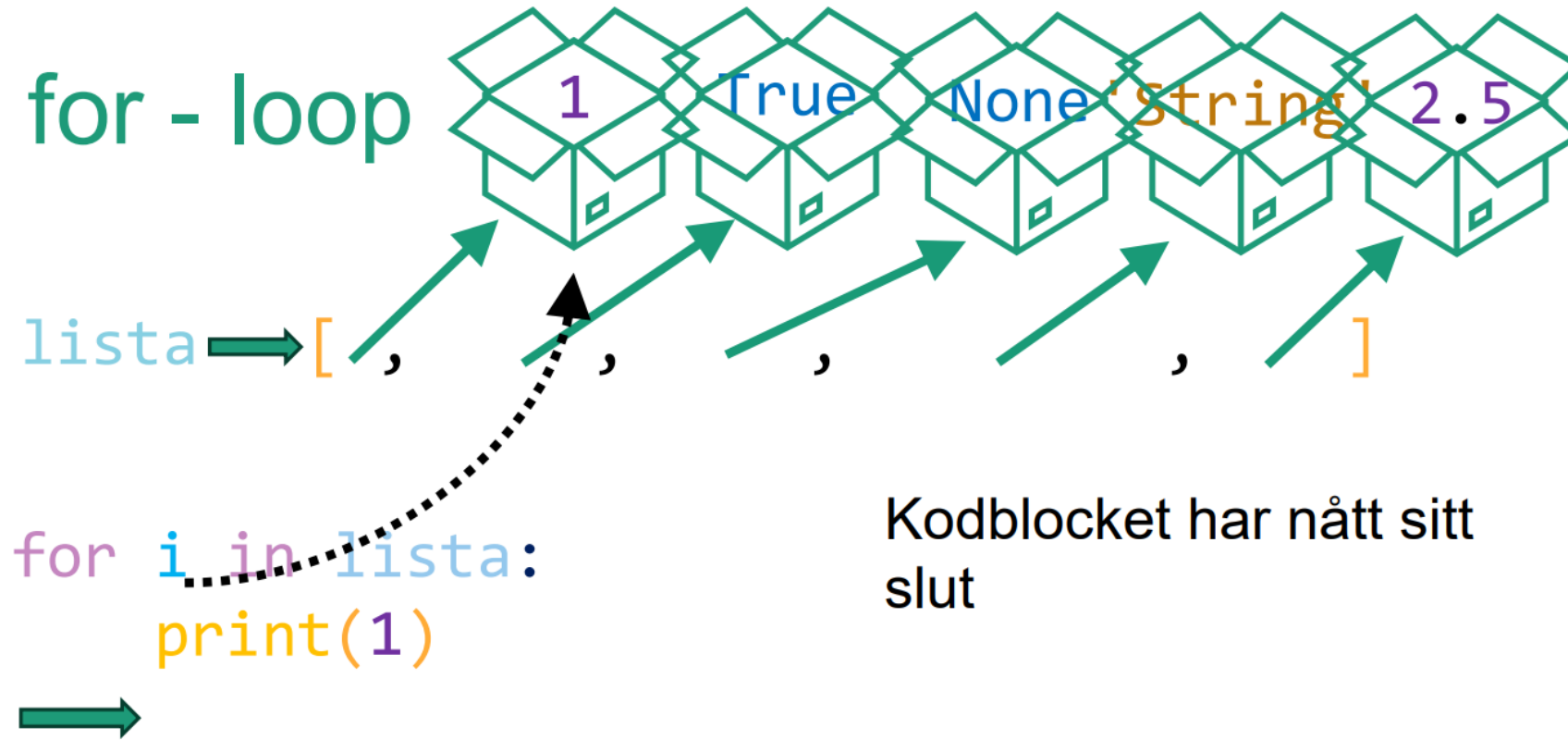
# for-loop



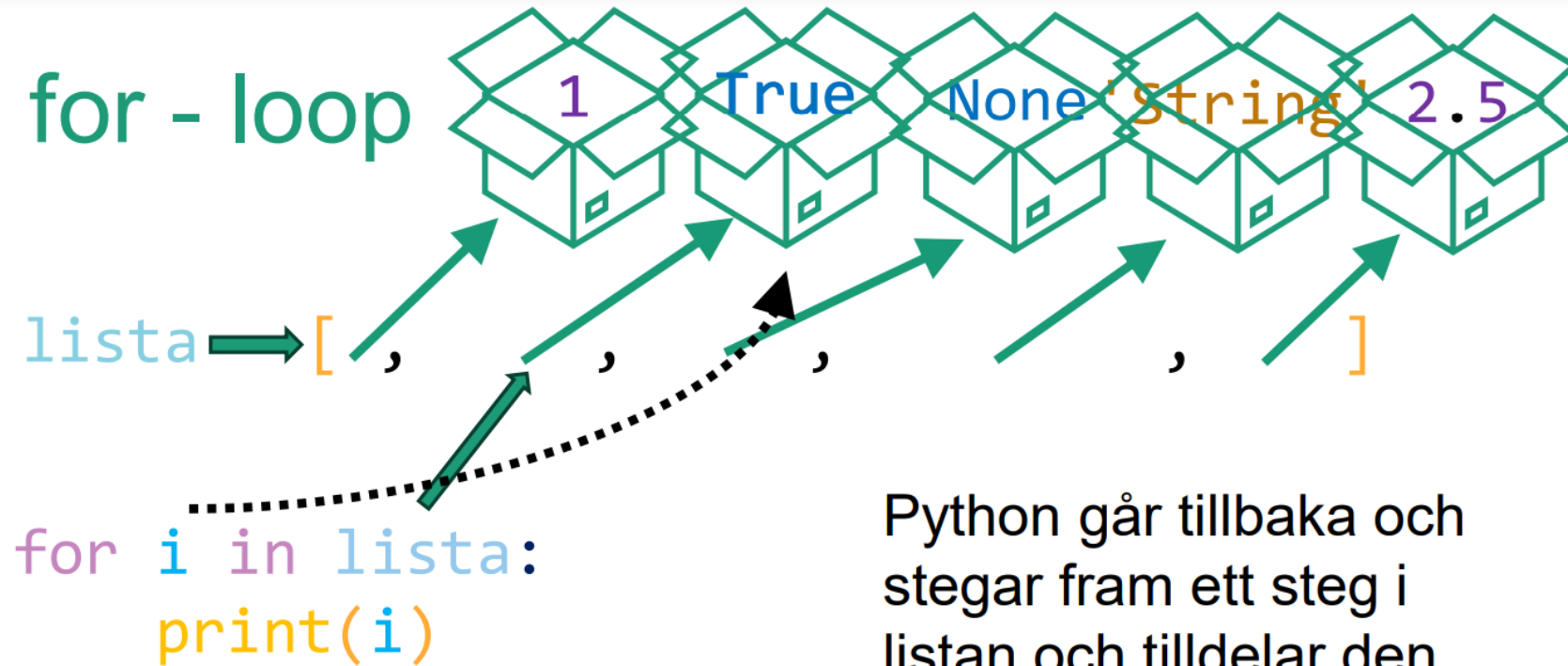
# for-loop



# for-loop

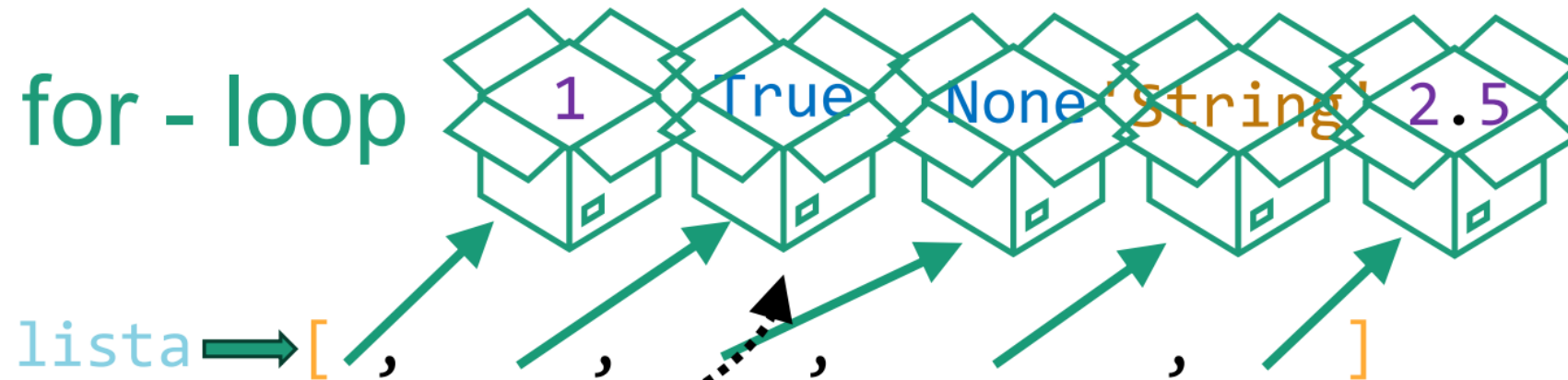


# for-loop



Python går tillbaka och stegar fram ett steg i listan och tilldelar den lokala variabeln det nya värdet!

# for-loop

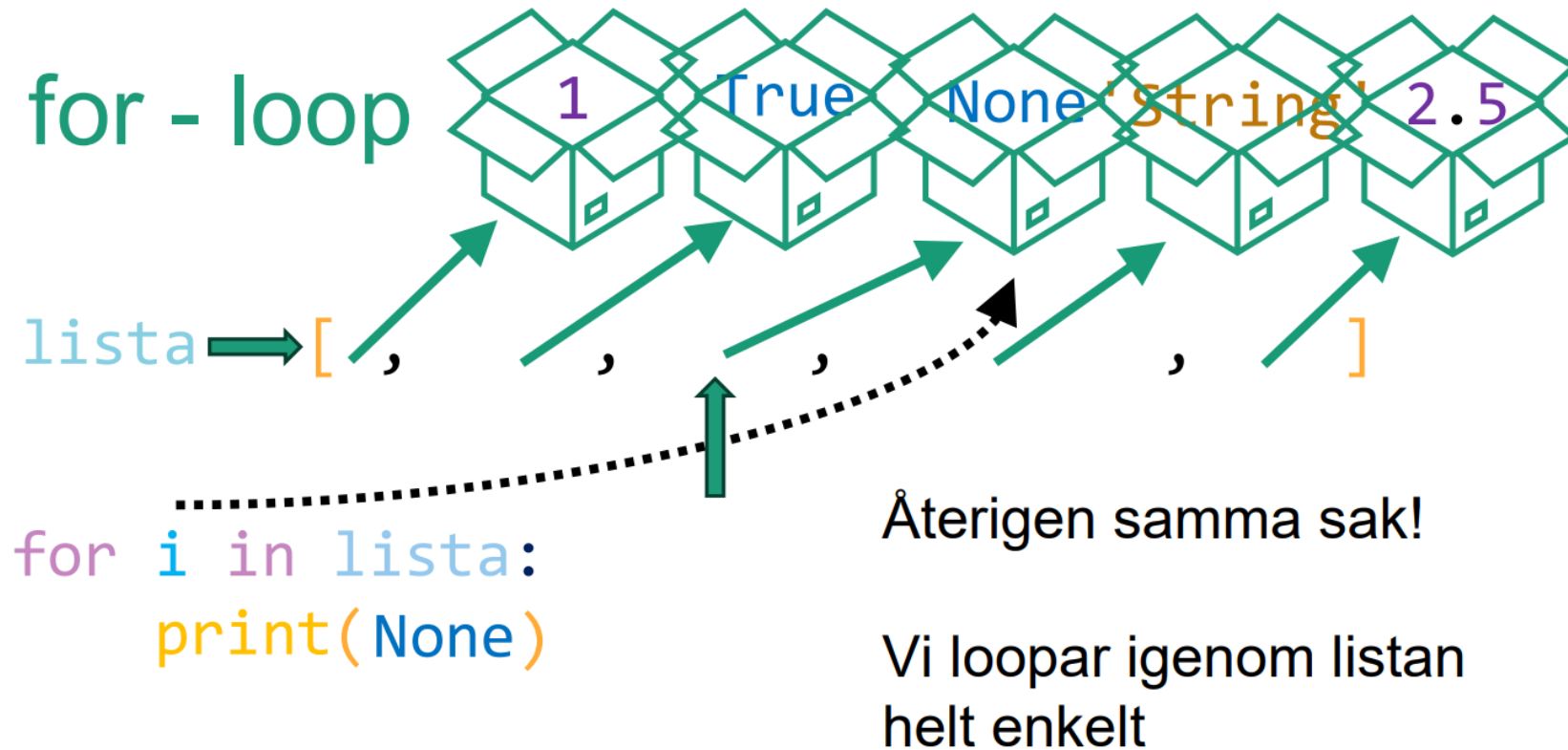


```
for i in lista:  
    → print(True)
```

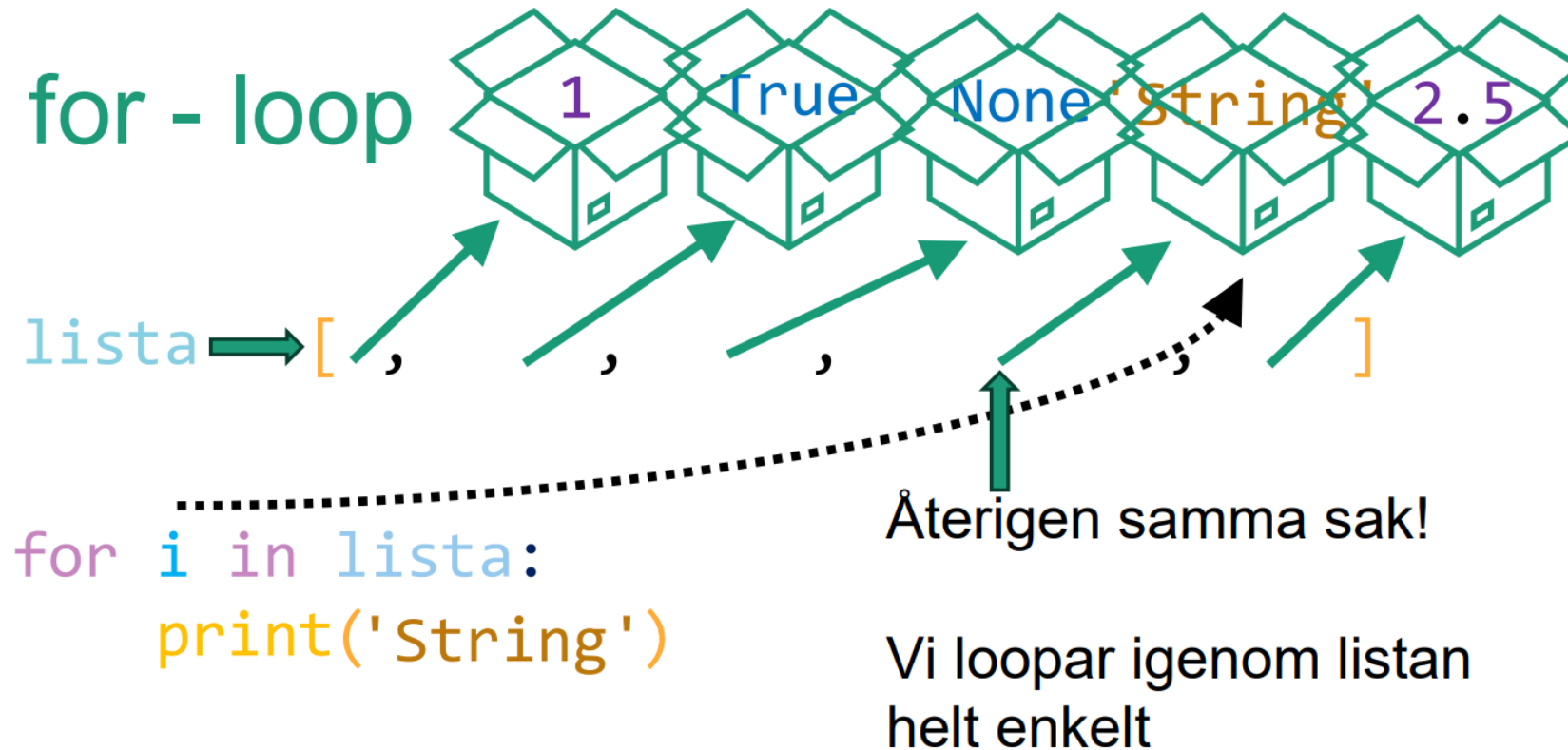
Kör kodblocket igen med  
nya nya värdet tilldelat till  
den lokala variabeln



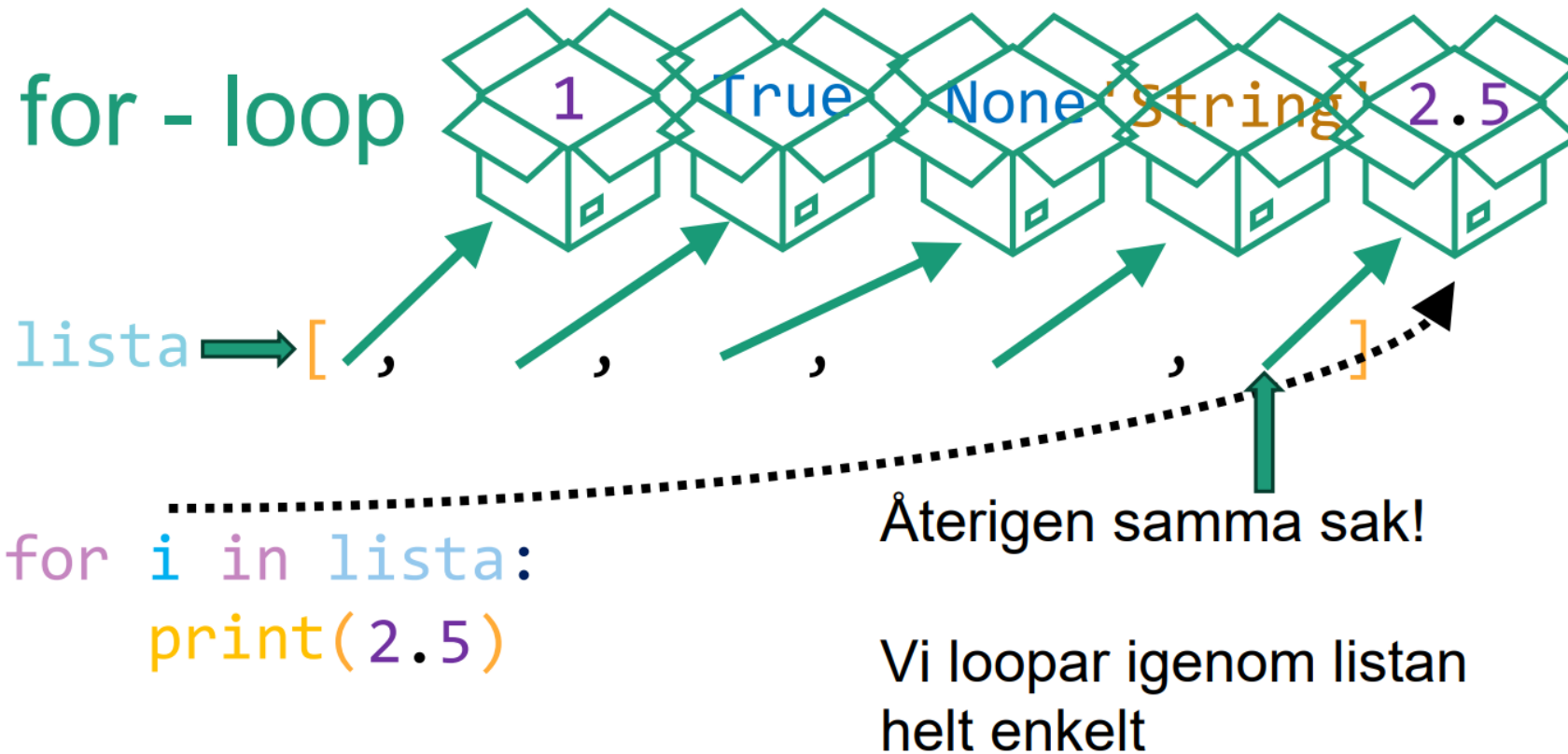
# for-loop



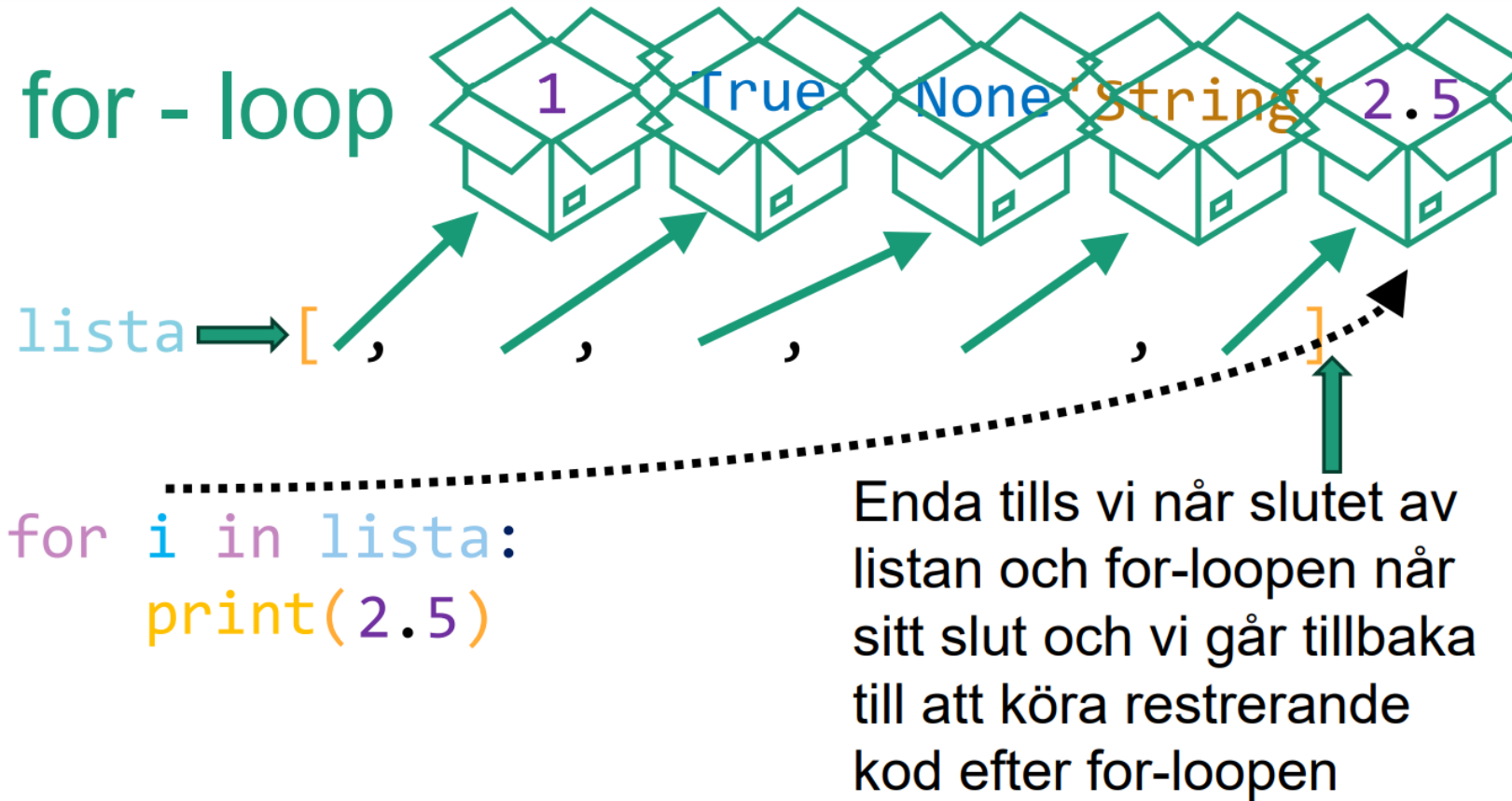
# for-loop



# for-loop



# for-loop



# while - loop

```
''' En while-loop består av först: nyckelordet 'while'  
följt av ett logiskt uttryck eller en boolesk datatyp  
följt av ett kolon. Därefter är all indenterad kod  
under den kod som kommer köras om uttrycket eller den  
booleska värdet är True!  
'''
```

```
while True:  
    print('Vi kommer köra för evigt!!')
```

# while - loop

```
count = 0
```



```
while count < 3:  
    print(count)  
    count += 1
```


# while - loop

count → 


→ 

```
while count < 3:  
    print(count)  
    count += 1
```

# while - loop


count → 

```
while 0 < 3:  
    print(count)  
    count += 1
```





# while - loop

count → 

while True:  
 print(count)  
 count += 1

# while - loop

count → 

```
while True:  
    → print(0)  
    count += 1
```

# while - loop

count → 

```
while True:  
    print(0)  
    → count += 1
```

# while - loop

count → 

```
while True:
```

```
    print(0)
```

```
    → count = count + 1
```

# while - loop

count → 

```
while True:
```

```
    print(0)
```

```
    → count = 0 + 1
```

# while - loop

count → 

```
while True:  
    print(0)  
    → count = 1
```

# while - loop

count → 

```
while True:  
    print(0)  
    count = 1
```



# while - loop

count → 

→ 

```
while count < 3:  
    print(count)  
    count += 1
```



# while - loop

count → 

→ 

```
while 1 < 3:  
    print(count)  
    count += 1
```

# while - loop

count → 

→ 

```
while True:  
    print(count)  
    count += 1
```

# while - loop

count → 

```
while True:  
    → print(1)  
    count += 1
```

# while - loop

count → 

```
while True:
```

```
    print(1)
```

```
    → count += 1
```

# while - loop

count → 

→ 

```
while count < 3:  
    print(count)  
    count += 1
```

# while - loop

count → 

→ 

```
while 2 < 3:  
    print(count)  
    count += 1
```

# while - loop

count → 

```
while True:  
    → print(2)  
    count += 1
```

# while - loop

count → 

```
while True:  
    print(2)  
    → count += 1
```



# while - loop

count → 

→ 

```
while count < 3:  
    print(count)  
    count += 1
```

# while - loop

count → 

→ 

```
while 3 < 3:  
    print(count)  
    count += 1
```

# while - loop

count → 

→ 

```
while False:  
    print(count)  
    count += 1
```

# while - loop

count → 

```
while False:  
    print(count)  
    count += 1
```

➡ Vi fortsätter köra programmet men hoppar över while-loopen. D.v.s. vi kör inte koden inne i while- loopen något mer!!