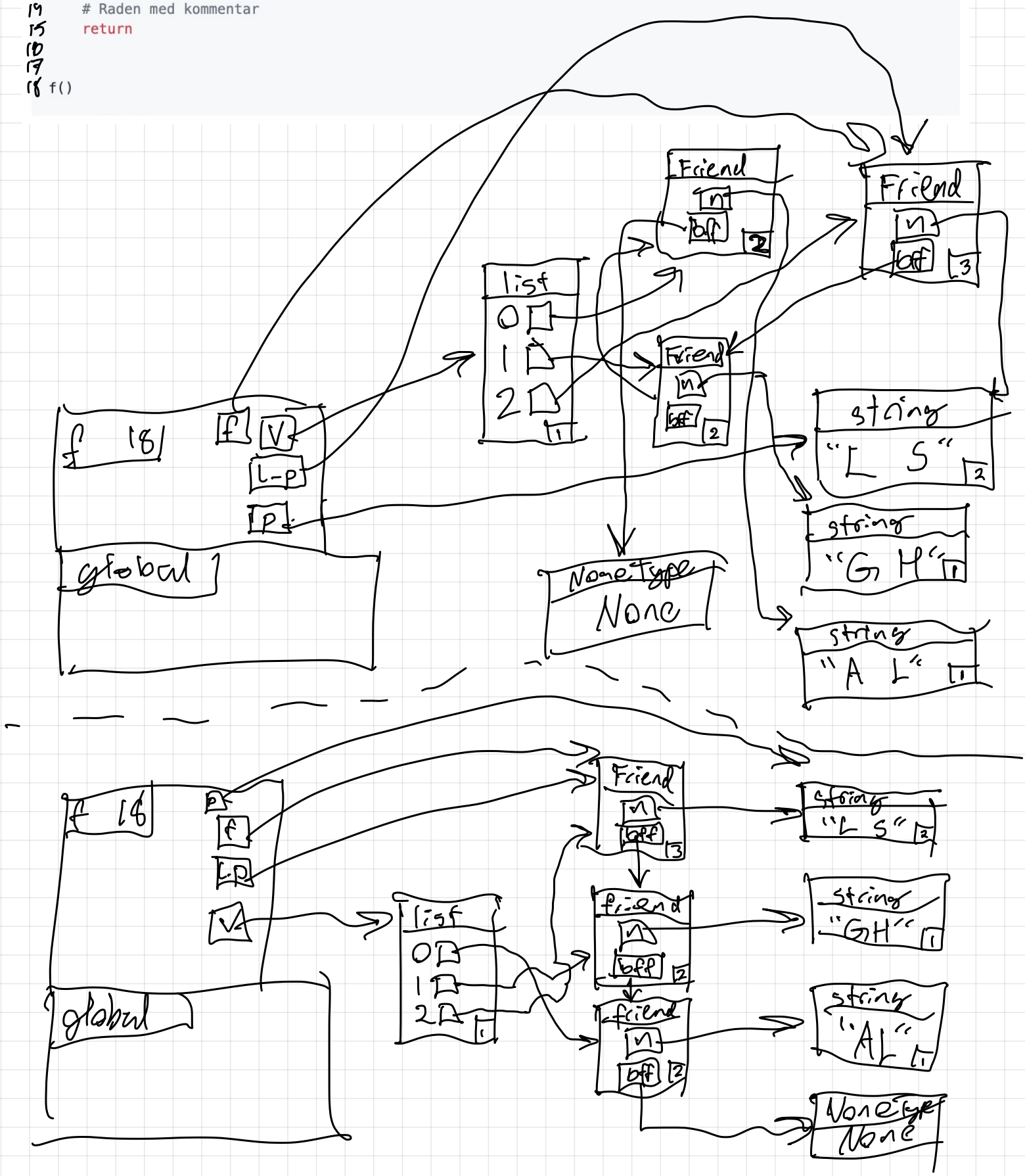
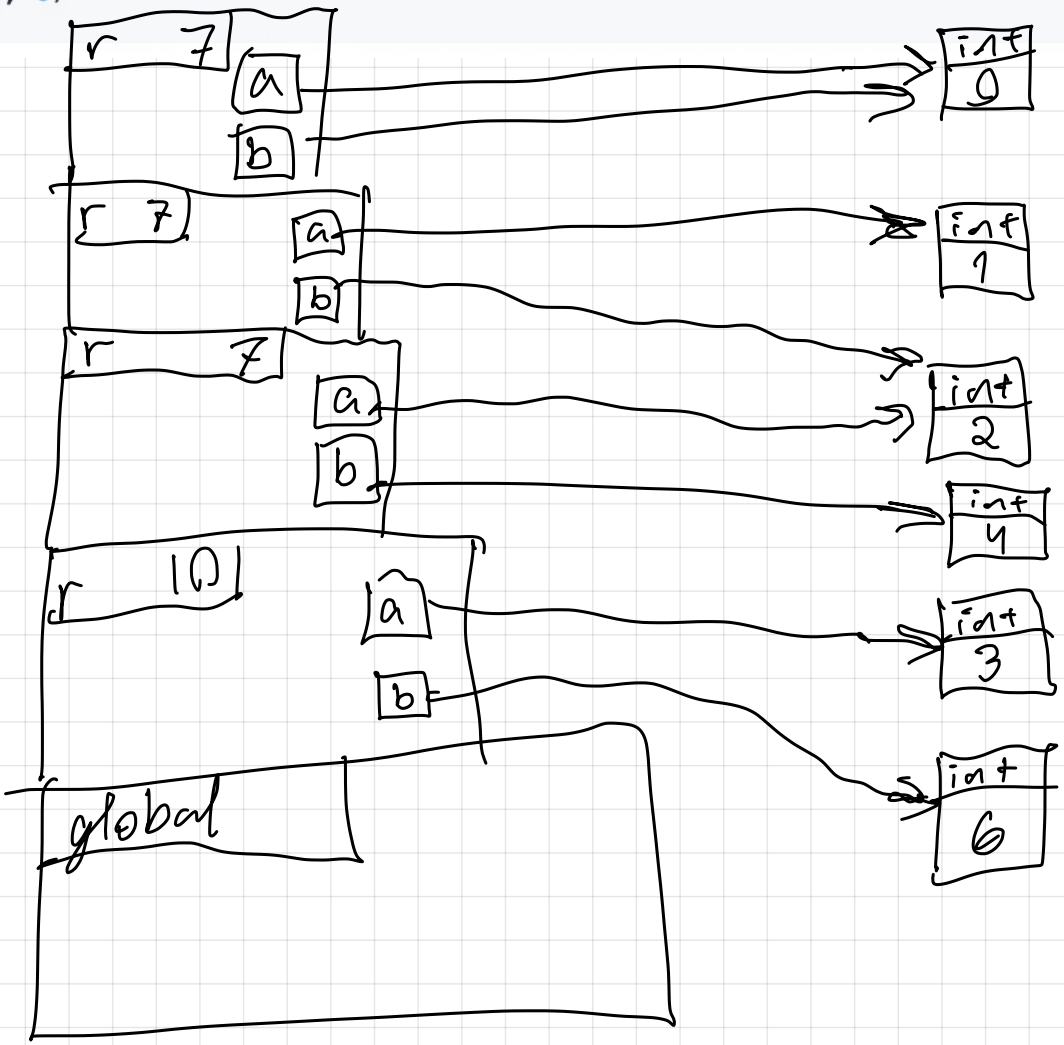
```python
class Friend:
    def __init__(self, name, bff):
        self.name = name
        self.bff = bff


def f():
    v = []
    last_person = None
    for person in ["Ada Lovelace", "Grace Hopper", "Lisa Su"]:
        f = Friend(person, last_person)
        last_person = f
        v.append(f)
    # Raden med kommentar
    return


f()
```



```python
def r(a, b):
    if a <= 0:
        # Raden med kommentar
        return 1
    if b <= 0:
        return 2
    return r(a-1, b-2)


r(3, 6)
```




return 1

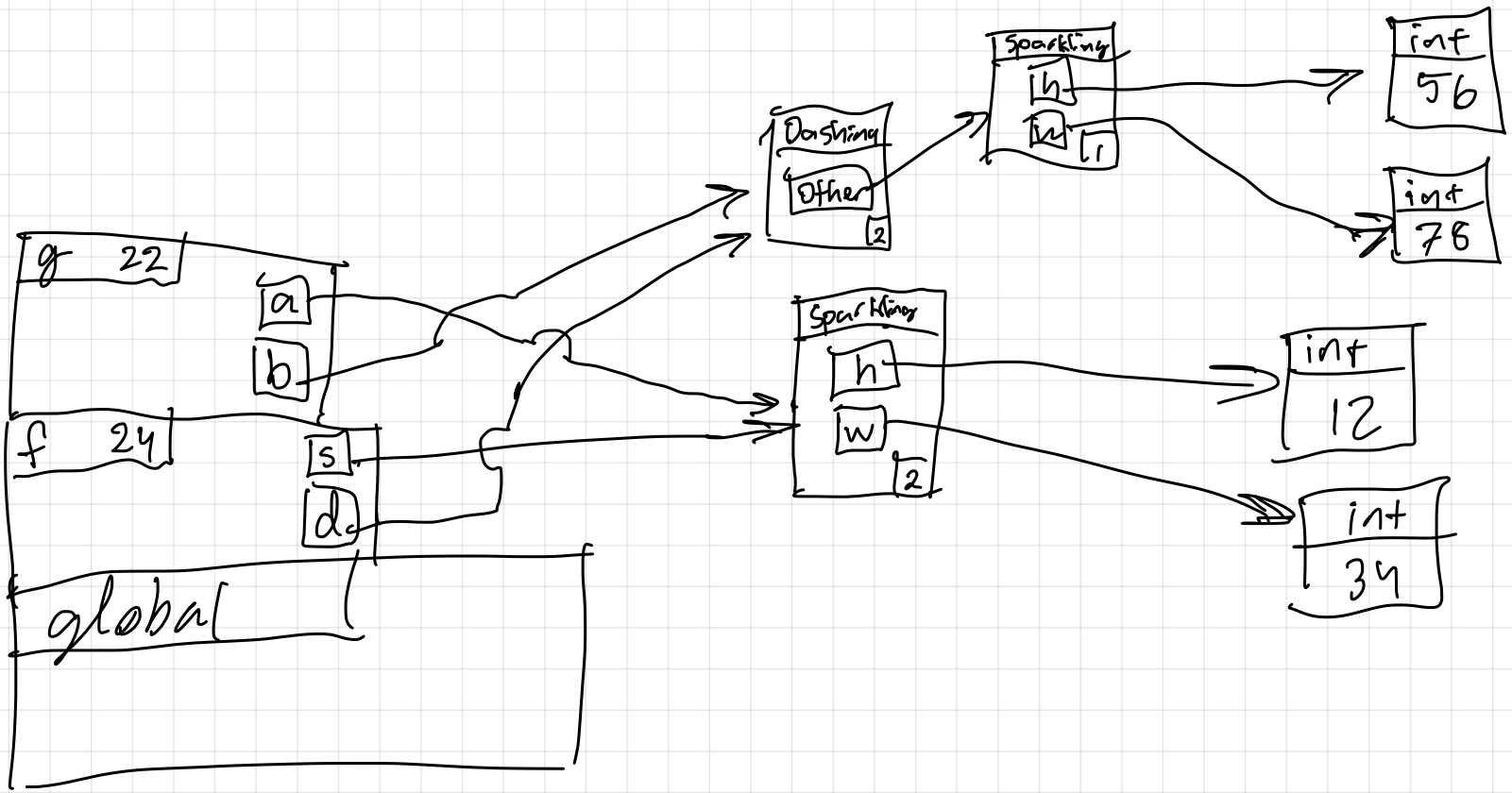
r(0,0)

r(1,2)

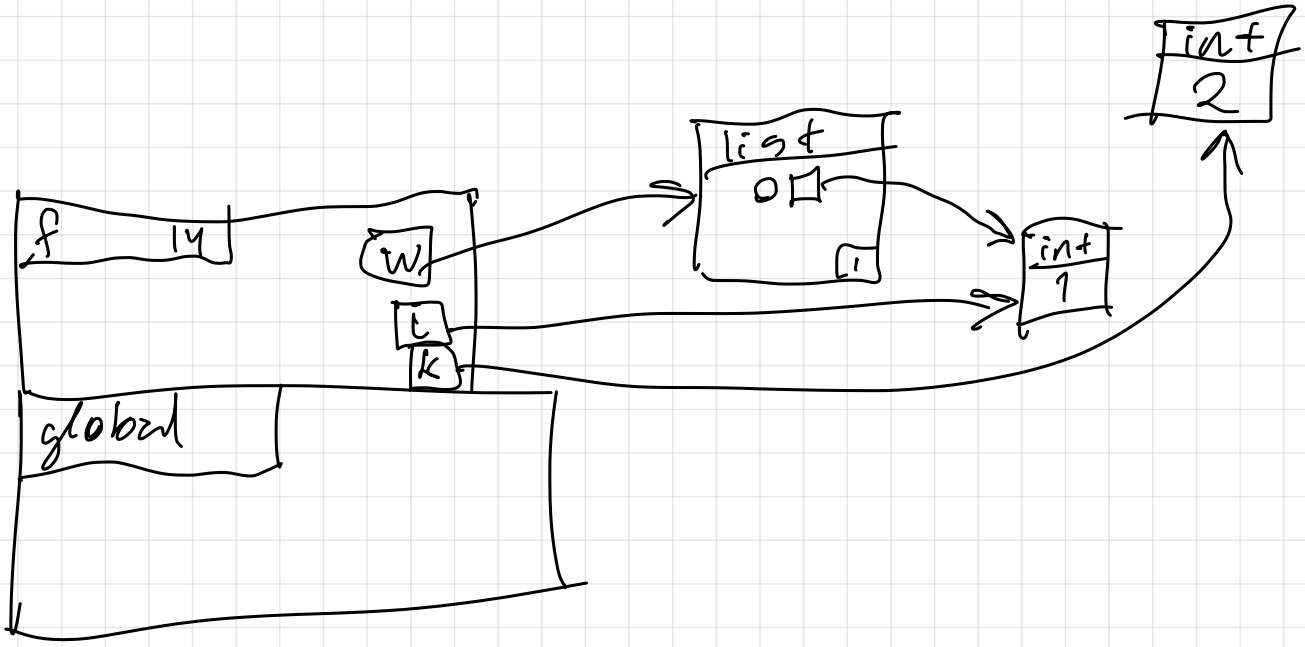
r(2,4)

r(3,6)

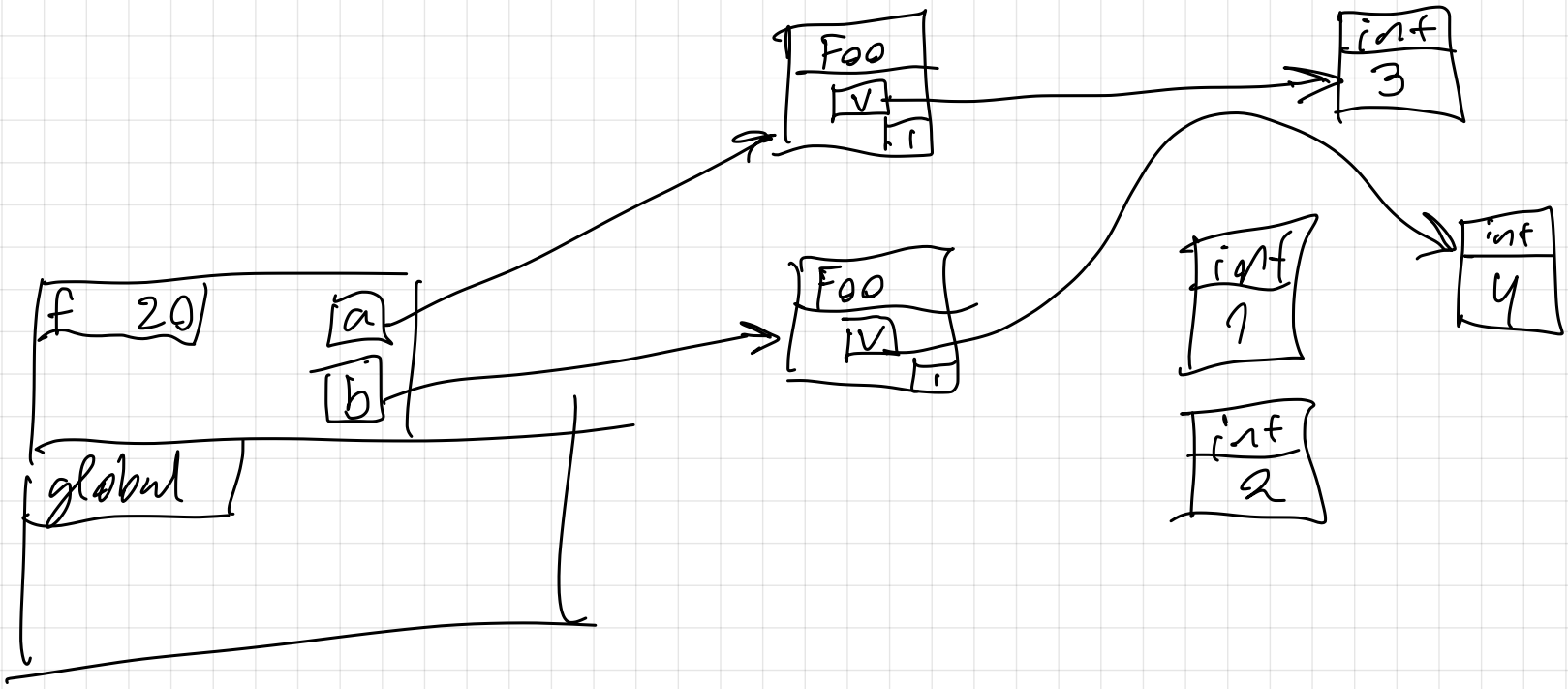
```python
class Sparkling:
    def __init__(self, height, width):
        self.height = height
        self.width = width


class Dashing:
    def __init__(self, other):
        self.other = other


def g(a, b):
    b.other = Sparkling(56, 78)
    # Raden med kommentar
    return


def f():
    s = Sparkling(12, 34)
    d = Dashing(s)
    g(s, d)


f()
```



```python
def g(v, i):
    v.append(i)
    i += 1
    return i


def f():
    w = []
    i = 1
    k = g(w, i)
    # Raden med kommentar


f()
```



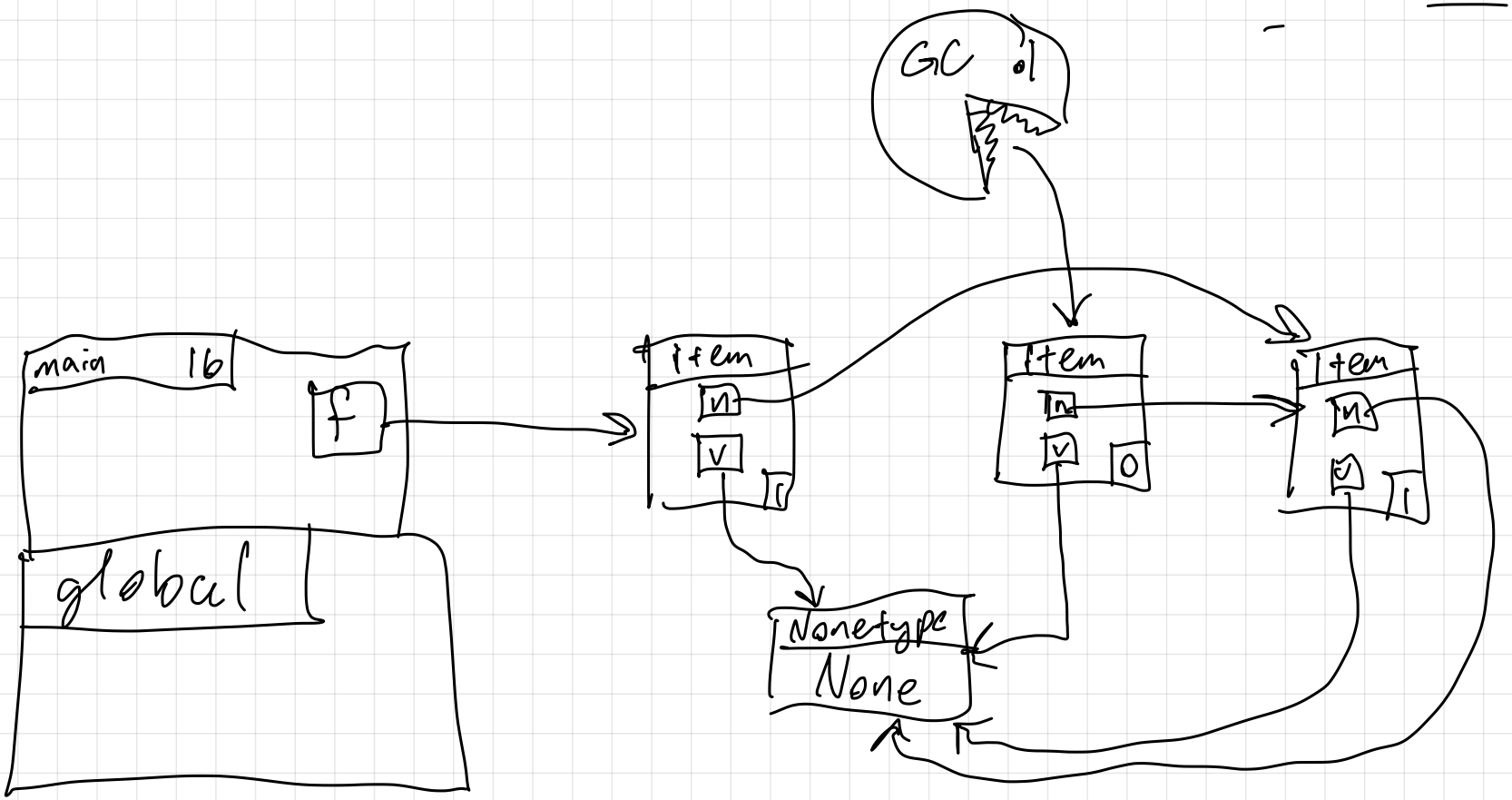
```python
class Foo:
    def __init__(self, value):
        self.value = value


def g(a, b):
    a.value = 3
    b.value = 4
    return


def f():
    a = Foo(1)
    b = Foo(2)
    g(a, b)
    # Raden med kommentar
    return


f()
```



9. Från exempeltenta:

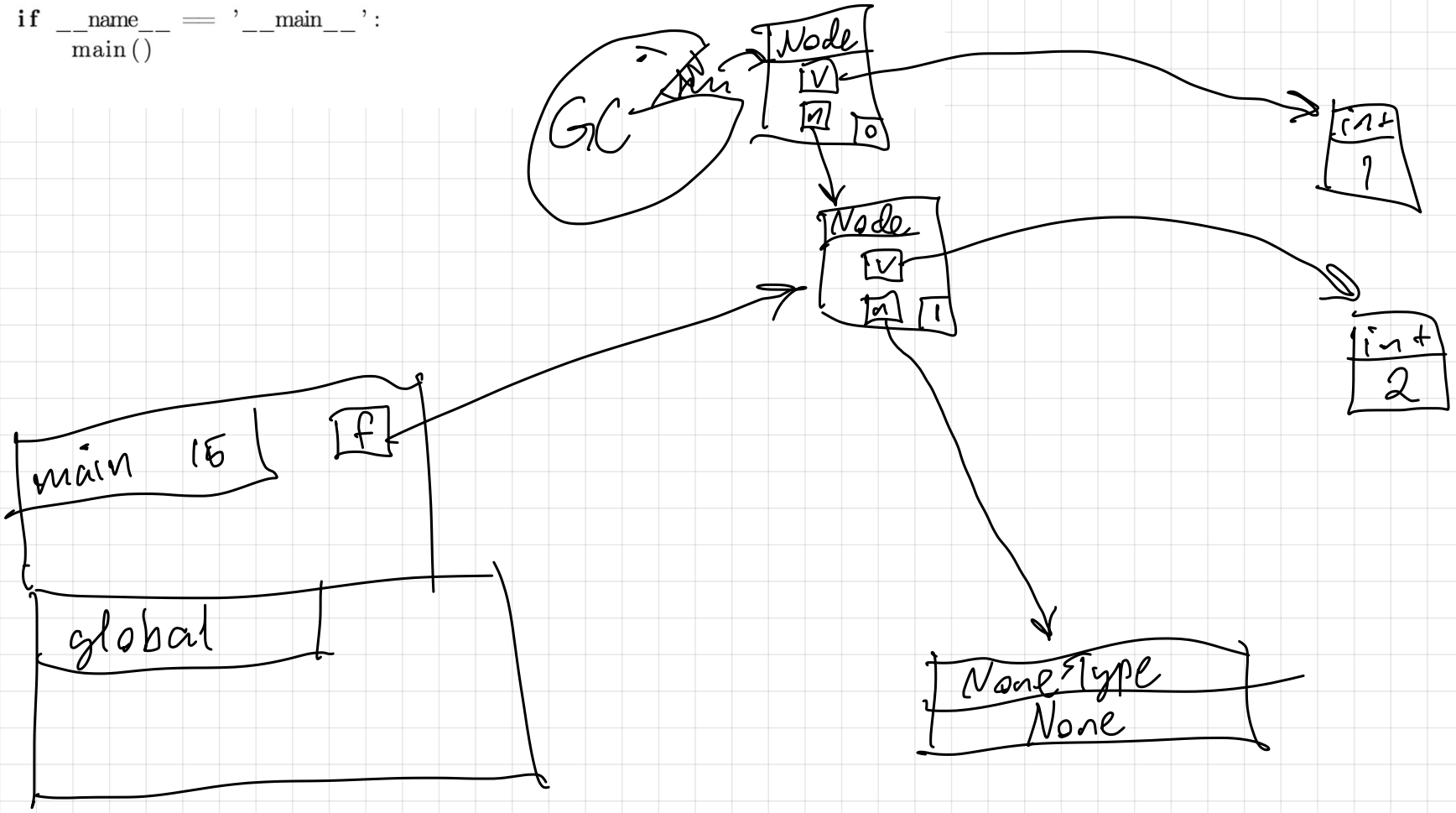
```
1. class Item:
2.     """A singly linked list."""
3.     def __init__(self, value=None, next=None):
4.         self.value = value
5.         self.next = next
6.
7.
8. def main():
9.     first = Item()
10.    first.next = Item()
11.    first.next.next = Item()
12.    first.next = first.next.next
13.    # Rita hur minnet ser ut här
14.
15.
16. main()
```

Class Node:
pass



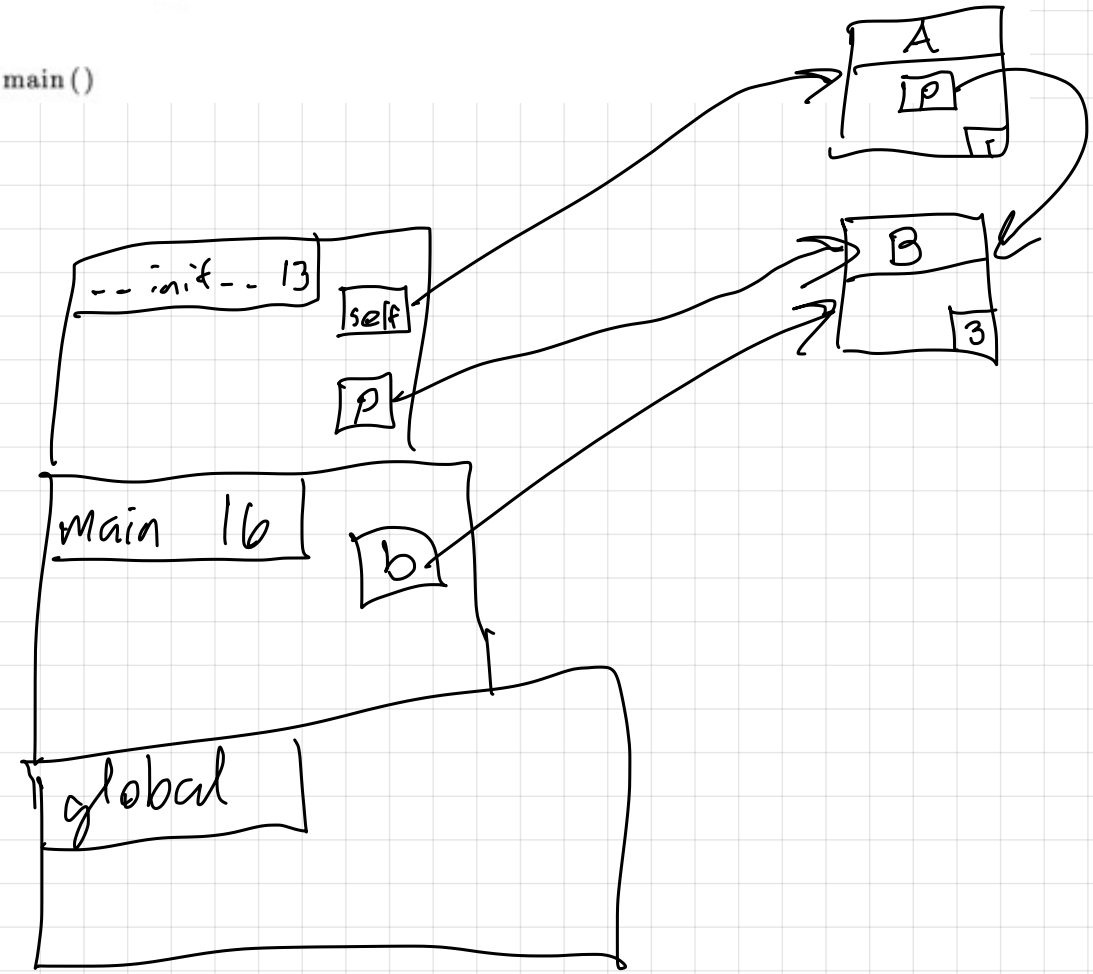
30. Rita ett låd- och pildiagram över minnet då programkörningen når den kommenterade raden.

```
1  class Node:
2      def __init__(self, v, n=None):
3          self.v = v
4          self.n = n
5
6
7  def main():
8      first = Node(2)
9      first = Node(1, first)
10     first = first.n
11     # Den kommenterade raden.
12
13
14 if __name__ == '__main__':
15     main()
```



28. Rita ett låd- och pildiagram över minnet då programkörningen når den kommenterade raden.

```
1  class A:
2      def __init__(self, parent):
3          self.parent = parent
4          # Den kommenterade raden.
5
6
7  class B:
8      pass
9
10
11 def main():
12     b = B()
13     a = A(b)
14
15
16 main()
```



# Vad tar GC inte? (Vad behöver inte referensräkning?)

- Funktioner i stacken
- Integers i intervallet [-5, 255]
- Booleans: True och False
- NoneType: None

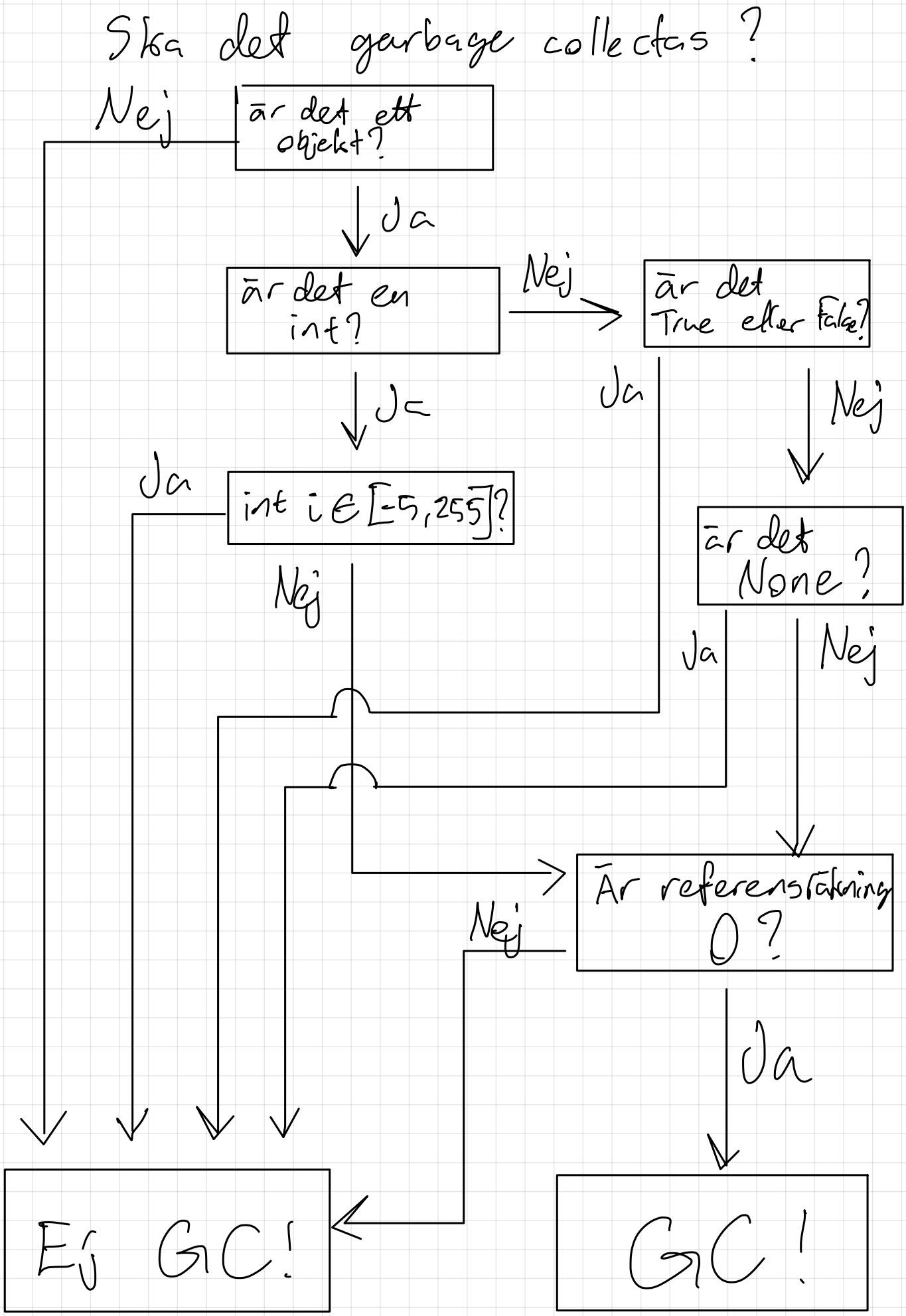
} (Finns alltid i minnet

- Pekare (pilar)

---

# Vad tar GC (inte komplett lista)?

- Listor
- Egenskapade objekt
- Strings
- tuples
- Dictionaries


Ska det garbage collectas?
Nej
är det ett objekt?
Ja
är det en int?
Nej
är det True eller False?
Ja
Ja
Nej
Ja
int i ∈ [-5, 255]?
Nej
är det None?
Ja
Nej
Är referensräkning 0?
Nej
Ja
Ej GC!
GC!