8 Queens

One solution → All solutions ?

```
def placequeen(i):
    for all positions (i,c):
        if (i,c) is free:
            if i == 8:
                return(True)
            try = placequeen(i+1)
            if try:
                return(True)
            else
return(False)
```

← define board etc global

← in terms of row, col, lrdiag, rldiag

← update board

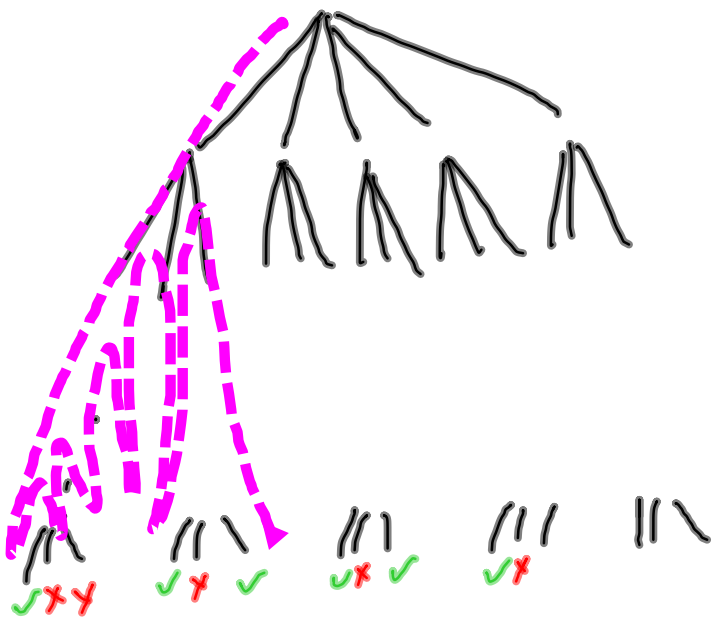
← undo (i,c) move, update board

global
board
row
col
lrdiag
rldiag

Report all solutions?

Print, or accumulate in a list, or ..

Record this solution and keep going



After successfully placing (i,c)

try to place queen r+1

generate all valid solns, if any, for (i,c)

try (i,c+1)

```
:
if (i,c) is free:
    update board
    if i==8:
        reword solution
    else
        placequeen(i+1)
    undo (i,c), update board
```

Global and local variables in Python

```
def f():
    y = x
    print(y)

x = 17
f()
```

```
def g():
    y = x
    print(y)

x = 12

x = 17
g()
```

Error: x has no value

makes this x local to g()

```
def h():
    global x
    y = x
    print(y)

x = 12

x = 17
h()
print(x)
```

17

12

Python allows nested function defns

```
def f():
    def g():
        =
    def h():
        =
```

nonlocal

local name

global name

share a name within f(), g(), h() ?

Python allows another qualifier

nonlocal

nonlocal: immediately surrounding level

global: outermost level

```
def f()
    def g()
        nonlocal x
        x=x+20

    def h()
        global x
        x=x+30
x=5; g(); h(); print(x)
```

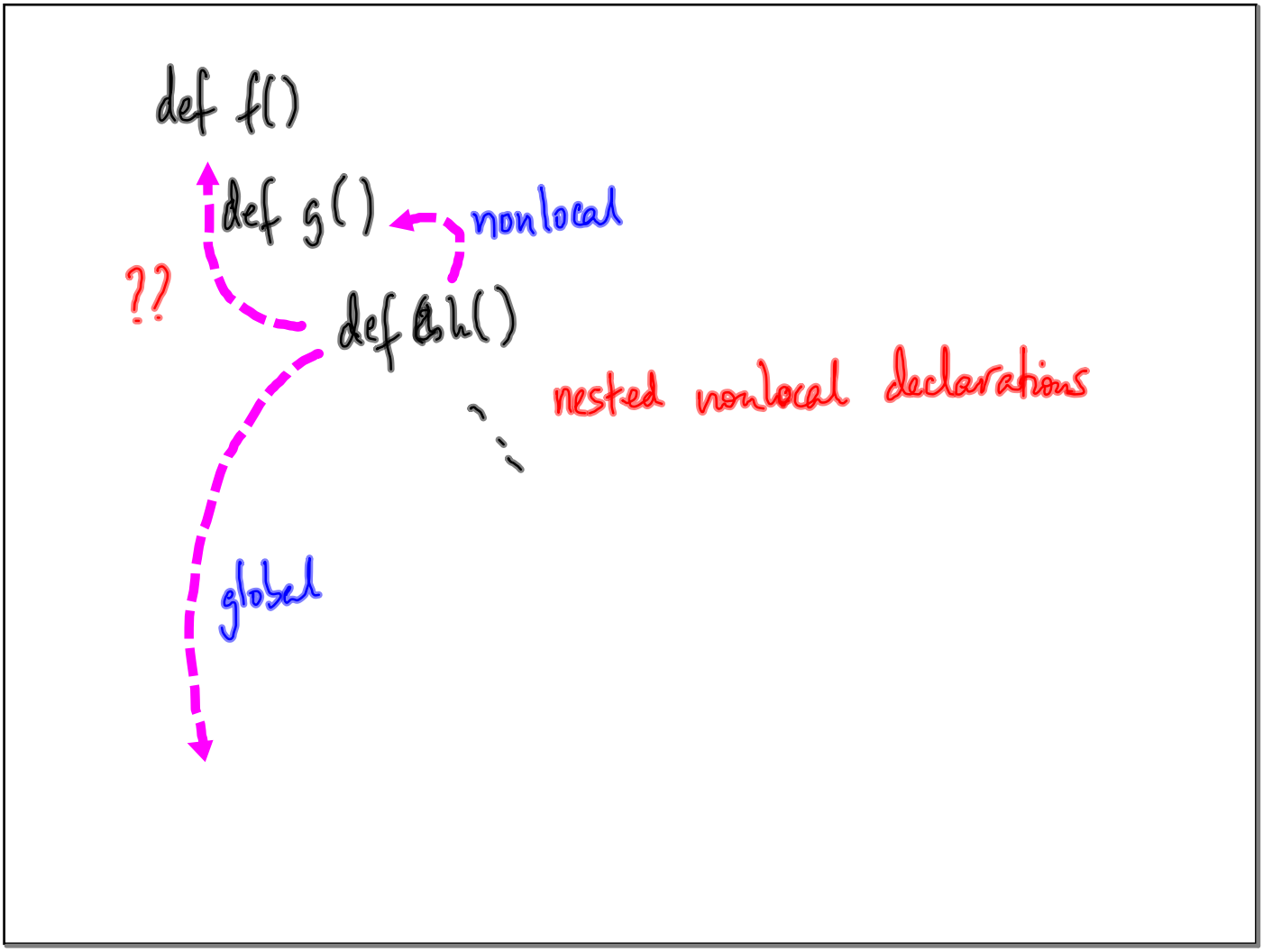
x local to f()

x is 25 (updated by g())

global x

```
x=10
f()
print(x)
```

x is 40 (updated by h())


def f()
def g()
nonlocal
??
def h()
nested nonlocal declarations
global

Recall how we solved all solutions of

8 queens

A nonoptimal soln

```
for all q1 at 1 to 8:
    for all q2 at 1 to 8:
        .
        .
        .
            if [q1, q2, . , q8] is a legal soln:
                record it
```

In some situations, it is useful to adopt the following approach

generate a possible configuration

test it

One instance of this is to generate all permutations of a sequence in dictionary order

$x_1 < x_2 < x_3 \; \cdots \; < x_n$

16 23 38 39 41 54 67 68 72 81

Smallest permutation — Ascending order

Biggest permutation — Descending order

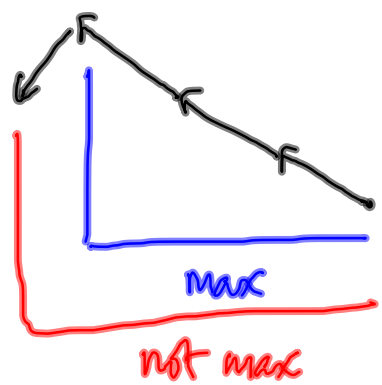
38 41 16 72 81 67 54 39 68 23

What comes next?

Identify

Prefix	Tail (smallest)
Left untouched	Can be increased i.e. not in desc. order

To identify the tail



Tail is

38 86 62 51 12

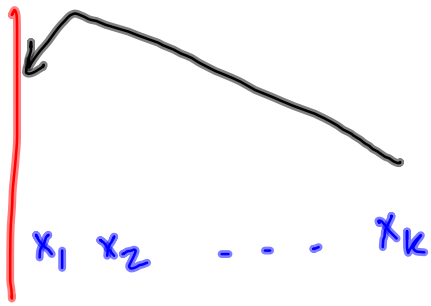
Next arrangement of tail ?

Change 38 to smallest larger value 51

51 12 38 62 86

smallest tail for 51

Work backwards from right to identify tail



Exchange x_1 with $x_j > x_1$ s.t. x_j is smallest such

x_j $\underline{x_2 \cdots x_{j-1}\ x_1\ x_{j+1} \cdots x_k}$ ← again in desc. order

sort ascending

= reverse!

x_j x_k x_{k-1} ... x_{j+1} x_1 x_{j-1} .. x_2