Programming and Data Structures with Python

Lecture 24, 15 Nov 2022

8 queens, passing the board

In [1]:
```
def initialize(board,n):
  for key in ['queen','row','col','nwtose','swtone']:
    board[key] = {}
  for i in range(n):
    board['queen'][i] = -1
    board['row'][i] = 0
    board['col'][i] = 0
  for i in range(-(n-1),n):
    board['nwtose'][i] = 0
  for i in range(2*n-1):
    board['swtone'][i] = 0

def printboard(board):
  for row in sorted(board['queen'].keys()):
    print((row,board['queen'][row]))

def free(i,j,board):
  return(board['row'][i] == 0 and board['col'][j] == 0 and
         board['nwtose'][j-i] == 0 and board['swtone'][j+i] == 0)

def addqueen(i,j,board):
  board['queen'][i] = j
  board['row'][i] = 1
  board['col'][j] = 1
  board['nwtose'][j-i] = 1
  board['swtone'][j+i] = 1

def undoqueen(i,j,board):
  board['queen'][i] = -1
  board['row'][i] = 0
  board['col'][j] = 0
  board['nwtose'][j-i] = 0
  board['swtone'][j+i] = 0

def placequeen(i,board):
  n = len(board['queen'].keys())
  for j in range(n):
    if free(i,j,board):
      addqueen(i,j,board)
      if i == n-1:
        return(True)
      else:
        extendsoln = placequeen(i+1,board)
      if extendsoln:
        return(True)
      else:
        undoqueen(i,j,board)
  else:
    return(False)

board = {}
n = int(input("How many queens? "))
initialize(board,n)
if placequeen(0,board):
  printboard(board)
else:
  print("no solution")
```

```
How many queens? 7
(0, 0)
(1, 2)
(2, 4)
(3, 6)
(4, 1)
(5, 3)
(6, 5)
```

8 queens, solution printed nicely

In [2]:
```
def initialize(board,n):
  for key in ['queen','row','col','nwtose','swtone']:
    board[key] = {}
  for i in range(n):
    board['queen'][i] = -1
    board['row'][i] = 0
    board['col'][i] = 0
  for i in range(-(n-1),n):
    board['nwtose'][i] = 0
  for i in range(2*n-1):
    board['swtone'][i] = 0

def printboard(board):
  n = len(board['queen'].keys())
  dashline = "----"*n + "-----"
  print(dashline)

  for row in sorted(board['queen'].keys()):
    pos = board['queen'][row]

    print("|", end="")

    for i in range(pos):
      print("   ", end="|")
    print(" Q ", end="|")
    for i in range(pos,n):
      print("   ", end="|")

    print()
    print(dashline)

def free(i,j,board):
  return(board['row'][i] == 0 and board['col'][j] == 0 and
         board['nwtose'][j-i] == 0 and board['swtone'][j+i] == 0)

def addqueen(i,j,board):
  board['queen'][i] = j
  board['row'][i] = 1
  board['col'][j] = 1
  board['nwtose'][j-i] = 1
  board['swtone'][j+i] = 1

def undoqueen(i,j,board):
  board['queen'][i] = -1
  board['row'][i] = 0
  board['col'][j] = 0
  board['nwtose'][j-i] = 0
  board['swtone'][j+i] = 0

def placequeen(i,board):
  n = len(board['queen'].keys())
  for j in range(n):
    if free(i,j,board):
      addqueen(i,j,board)
      if i == n-1:
        return(True)
      else:
        extendsoln = placequeen(i+1,board)
      if extendsoln:
        return(True)
      else:
        undoqueen(i,j,board)
  else:
    return(False)

board = {}
n = int(input("How many queens? "))
initialize(board,n)
if placequeen(0,board):
  printboard(board)
```

```
How many queens? 7
---------------------------------
| Q |   |   |   |   |   |   |   |
---------------------------------
|   |   | Q |   |   |   |   |   |
---------------------------------
|   |   |   |   | Q |   |   |   |
---------------------------------
|   |   |   |   |   |   | Q |   |
---------------------------------
|   | Q |   |   |   |   |   |   |
---------------------------------
|   |   |   | Q |   |   |   |   |
---------------------------------
|   |   |   |   |   | Q |   |   |
---------------------------------
```

8 queens, all solutions

In [3]:

```
def initialize(board,n):
  for key in ['queen','row','col','nwtose','swtone']:
    board[key] = {}
  for i in range(n):
    board['queen'][i] = -1
    board['row'][i] = 0
    board['col'][i] = 0
  for i in range(-(n-1),n):
    board['nwtose'][i] = 0
  for i in range(2*n-1):
    board['swtone'][i] = 0

def printboard(board):
  for row in sorted(board['queen'].keys()):
    print((row,board['queen'][row]),end=" ")
  print("")

def free(i,j,board):
  return(board['row'][i] == 0 and board['col'][j] == 0 and
         board['nwtose'][j-i] == 0 and board['swtone'][j+i] == 0)

def addqueen(i,j,board):
  board['queen'][i] = j
  board['row'][i] = 1
  board['col'][j] = 1
  board['nwtose'][j-i] = 1
  board['swtone'][j+i] = 1

def undoqueen(i,j,board):
  board['queen'][i] = -1
  board['row'][i] = 0
  board['col'][j] = 0
  board['nwtose'][j-i] = 0
  board['swtone'][j+i] = 0

def placequeen(i,board):
  n = len(board['queen'].keys())
  for j in range(n):
    if free(i,j,board):
      addqueen(i,j,board)
      if i == n-1:
        printboard(board)
      else:
        extendsoln = placequeen(i+1,board)
      undoqueen(i,j,board)

board = {}
n = int(input("How many queens? "))
initialize(board,n)
if placequeen(0,board):
  printboard(board)
```

```
How many queens? 8
(0, 0) (1, 4) (2, 7) (3, 5) (4, 2) (5, 6) (6, 1) (7, 3)
(0, 0) (1, 5) (2, 7) (3, 2) (4, 6) (5, 3) (6, 1) (7, 4)
(0, 0) (1, 6) (2, 3) (3, 5) (4, 7) (5, 1) (6, 4) (7, 2)
(0, 0) (1, 6) (2, 4) (3, 7) (4, 1) (5, 3) (6, 5) (7, 2)
(0, 1) (1, 3) (2, 5) (3, 7) (4, 2) (5, 0) (6, 6) (7, 4)
(0, 1) (1, 4) (2, 6) (3, 0) (4, 2) (5, 7) (6, 5) (7, 3)
(0, 1) (1, 4) (2, 6) (3, 3) (4, 0) (5, 7) (6, 5) (7, 2)
(0, 1) (1, 5) (2, 0) (3, 6) (4, 3) (5, 7) (6, 2) (7, 4)
(0, 1) (1, 5) (2, 7) (3, 2) (4, 0) (5, 3) (6, 6) (7, 4)
(0, 1) (1, 6) (2, 2) (3, 5) (4, 7) (5, 4) (6, 0) (7, 3)
(0, 1) (1, 6) (2, 4) (3, 7) (4, 0) (5, 3) (6, 5) (7, 2)
(0, 1) (1, 7) (2, 5) (3, 0) (4, 2) (5, 4) (6, 6) (7, 3)
(0, 2) (1, 0) (2, 6) (3, 4) (4, 7) (5, 1) (6, 3) (7, 5)
(0, 2) (1, 4) (2, 1) (3, 7) (4, 0) (5, 6) (6, 3) (7, 5)
(0, 2) (1, 4) (2, 1) (3, 7) (4, 5) (5, 3) (6, 6) (7, 0)
(0, 2) (1, 4) (2, 6) (3, 0) (4, 3) (5, 1) (6, 7) (7, 5)
(0, 2) (1, 4) (2, 7) (3, 3) (4, 0) (5, 6) (6, 1) (7, 5)
(0, 2) (1, 5) (2, 1) (3, 4) (4, 7) (5, 0) (6, 6) (7, 3)
(0, 2) (1, 5) (2, 1) (3, 6) (4, 0) (5, 3) (6, 7) (7, 4)
(0, 2) (1, 5) (2, 1) (3, 6) (4, 4) (5, 0) (6, 7) (7, 3)
(0, 2) (1, 5) (2, 3) (3, 0) (4, 7) (5, 4) (6, 6) (7, 1)
(0, 2) (1, 5) (2, 3) (3, 1) (4, 7) (5, 4) (6, 6) (7, 0)
(0, 2) (1, 5) (2, 7) (3, 0) (4, 3) (5, 6) (6, 4) (7, 1)
(0, 2) (1, 5) (2, 7) (3, 0) (4, 4) (5, 6) (6, 1) (7, 3)
(0, 2) (1, 5) (2, 7) (3, 1) (4, 3) (5, 0) (6, 6) (7, 4)
(0, 2) (1, 6) (2, 1) (3, 7) (4, 4) (5, 0) (6, 3) (7, 5)
(0, 2) (1, 6) (2, 1) (3, 7) (4, 5) (5, 3) (6, 0) (7, 4)
(0, 2) (1, 7) (2, 3) (3, 6) (4, 0) (5, 5) (6, 1) (7, 4)
(0, 3) (1, 0) (2, 4) (3, 7) (4, 1) (5, 6) (6, 2) (7, 5)
(0, 3) (1, 0) (2, 4) (3, 7) (4, 5) (5, 2) (6, 6) (7, 1)
(0, 3) (1, 1) (2, 4) (3, 7) (4, 5) (5, 0) (6, 2) (7, 6)
(0, 3) (1, 1) (2, 6) (3, 2) (4, 5) (5, 7) (6, 0) (7, 4)
(0, 3) (1, 1) (2, 6) (3, 2) (4, 5) (5, 7) (6, 4) (7, 0)
(0, 3) (1, 1) (2, 6) (3, 4) (4, 0) (5, 7) (6, 5) (7, 2)
(0, 3) (1, 1) (2, 7) (3, 4) (4, 6) (5, 0) (6, 2) (7, 5)
(0, 3) (1, 1) (2, 7) (3, 5) (4, 0) (5, 2) (6, 4) (7, 6)
(0, 3) (1, 5) (2, 0) (3, 4) (4, 1) (5, 7) (6, 2) (7, 6)
(0, 3) (1, 5) (2, 7) (3, 1) (4, 6) (5, 0) (6, 2) (7, 4)
(0, 3) (1, 5) (2, 7) (3, 2) (4, 0) (5, 6) (6, 4) (7, 1)
(0, 3) (1, 6) (2, 0) (3, 7) (4, 4) (5, 1) (6, 5) (7, 2)
(0, 3) (1, 6) (2, 2) (3, 7) (4, 1) (5, 4) (6, 0) (7, 5)
(0, 3) (1, 6) (2, 4) (3, 1) (4, 5) (5, 0) (6, 2) (7, 7)
(0, 3) (1, 6) (2, 4) (3, 2) (4, 0) (5, 5) (6, 7) (7, 1)
(0, 3) (1, 7) (2, 0) (3, 2) (4, 5) (5, 1) (6, 6) (7, 4)
(0, 3) (1, 7) (2, 0) (3, 4) (4, 6) (5, 1) (6, 5) (7, 2)
(0, 3) (1, 7) (2, 4) (3, 2) (4, 0) (5, 6) (6, 1) (7, 5)
(0, 4) (1, 0) (2, 3) (3, 5) (4, 7) (5, 1) (6, 6) (7, 2)
```

```
(0, 4) (1, 0) (2, 7) (3, 3) (4, 1) (5, 6) (6, 2) (7, 5)
(0, 4) (1, 0) (2, 7) (3, 5) (4, 2) (5, 6) (6, 1) (7, 3)
(0, 4) (1, 1) (2, 3) (3, 5) (4, 7) (5, 2) (6, 0) (7, 6)
(0, 4) (1, 1) (2, 3) (3, 6) (4, 2) (5, 7) (6, 5) (7, 0)
(0, 4) (1, 1) (2, 5) (3, 0) (4, 6) (5, 3) (6, 7) (7, 2)
(0, 4) (1, 1) (2, 7) (3, 0) (4, 3) (5, 6) (6, 2) (7, 5)
(0, 4) (1, 2) (2, 0) (3, 5) (4, 7) (5, 1) (6, 3) (7, 6)
(0, 4) (1, 2) (2, 0) (3, 6) (4, 1) (5, 7) (6, 5) (7, 3)
(0, 4) (1, 2) (2, 7) (3, 3) (4, 6) (5, 0) (6, 5) (7, 1)
(0, 4) (1, 6) (2, 0) (3, 2) (4, 7) (5, 5) (6, 3) (7, 1)
(0, 4) (1, 6) (2, 0) (3, 3) (4, 1) (5, 7) (6, 5) (7, 2)
(0, 4) (1, 6) (2, 1) (3, 3) (4, 7) (5, 0) (6, 2) (7, 5)
(0, 4) (1, 6) (2, 1) (3, 5) (4, 2) (5, 0) (6, 3) (7, 7)
(0, 4) (1, 6) (2, 1) (3, 5) (4, 2) (5, 0) (6, 7) (7, 3)
(0, 4) (1, 6) (2, 3) (3, 0) (4, 2) (5, 7) (6, 5) (7, 1)
(0, 4) (1, 7) (2, 3) (3, 0) (4, 2) (5, 5) (6, 1) (7, 6)
(0, 4) (1, 7) (2, 3) (3, 0) (4, 6) (5, 1) (6, 5) (7, 2)
(0, 5) (1, 0) (2, 4) (3, 1) (4, 7) (5, 2) (6, 6) (7, 3)
(0, 5) (1, 1) (2, 6) (3, 0) (4, 2) (5, 4) (6, 7) (7, 3)
(0, 5) (1, 1) (2, 6) (3, 0) (4, 3) (5, 7) (6, 4) (7, 2)
(0, 5) (1, 2) (2, 0) (3, 6) (4, 4) (5, 7) (6, 1) (7, 3)
(0, 5) (1, 2) (2, 0) (3, 7) (4, 3) (5, 1) (6, 6) (7, 4)
(0, 5) (1, 2) (2, 0) (3, 7) (4, 4) (5, 1) (6, 3) (7, 6)
(0, 5) (1, 2) (2, 4) (3, 6) (4, 0) (5, 3) (6, 1) (7, 7)
(0, 5) (1, 2) (2, 4) (3, 7) (4, 0) (5, 3) (6, 1) (7, 6)
(0, 5) (1, 2) (2, 6) (3, 1) (4, 3) (5, 7) (6, 0) (7, 4)
(0, 5) (1, 2) (2, 6) (3, 1) (4, 7) (5, 4) (6, 0) (7, 3)
(0, 5) (1, 2) (2, 6) (3, 3) (4, 0) (5, 7) (6, 1) (7, 4)
(0, 5) (1, 3) (2, 0) (3, 4) (4, 7) (5, 1) (6, 6) (7, 2)
(0, 5) (1, 3) (2, 1) (3, 7) (4, 4) (5, 6) (6, 0) (7, 2)
(0, 5) (1, 3) (2, 6) (3, 0) (4, 2) (5, 4) (6, 1) (7, 7)
(0, 5) (1, 3) (2, 6) (3, 0) (4, 7) (5, 1) (6, 4) (7, 2)
(0, 5) (1, 7) (2, 1) (3, 3) (4, 0) (5, 6) (6, 4) (7, 2)
(0, 6) (1, 0) (2, 2) (3, 7) (4, 5) (5, 3) (6, 1) (7, 4)
(0, 6) (1, 1) (2, 3) (3, 0) (4, 7) (5, 4) (6, 2) (7, 5)
(0, 6) (1, 1) (2, 5) (3, 2) (4, 0) (5, 3) (6, 7) (7, 4)
(0, 6) (1, 2) (2, 0) (3, 5) (4, 7) (5, 4) (6, 1) (7, 3)
(0, 6) (1, 2) (2, 7) (3, 1) (4, 4) (5, 0) (6, 5) (7, 3)
(0, 6) (1, 3) (2, 1) (3, 4) (4, 7) (5, 0) (6, 2) (7, 5)
(0, 6) (1, 3) (2, 1) (3, 7) (4, 5) (5, 0) (6, 2) (7, 4)
(0, 6) (1, 4) (2, 2) (3, 0) (4, 5) (5, 7) (6, 1) (7, 3)
(0, 7) (1, 1) (2, 3) (3, 0) (4, 6) (5, 4) (6, 2) (7, 5)
(0, 7) (1, 1) (2, 4) (3, 2) (4, 0) (5, 6) (6, 3) (7, 5)
(0, 7) (1, 2) (2, 0) (3, 5) (4, 1) (5, 4) (6, 6) (7, 3)
(0, 7) (1, 3) (2, 0) (3, 2) (4, 5) (5, 1) (6, 6) (7, 4)
```

8 queens, using a global board

In [4]:
```
def initialize(n):
  for key in ['queen','row','col','nwtose','swtone']:
    board[key] = {}
  for i in range(n):
    board['queen'][i] = -1
    board['row'][i] = 0
    board['col'][i] = 0
  for i in range(-(n-1),n):
    board['nwtose'][i] = 0
  for i in range(2*n-1):
    board['swtone'][i] = 0

def printboard():
  for row in sorted(board['queen'].keys()):
    print((row,board['queen'][row]))

def free(i,j):
  return(board['row'][i] == 0 and board['col'][j] == 0 and
         board['nwtose'][j-i] == 0 and board['swtone'][j+i] == 0)

def addqueen(i,j):
  board['queen'][i] = j
  board['row'][i] = 1
  board['col'][j] = 1
  board['nwtose'][j-i] = 1
  board['swtone'][j+i] = 1

def undoqueen(i,j):
  board['queen'][i] = -1
  board['row'][i] = 0
  board['col'][j] = 0
  board['nwtose'][j-i] = 0
  board['swtone'][j+i] = 0

def placequeen(i):
  n = len(board['queen'].keys())
  for j in range(n):
    if free(i,j):
      addqueen(i,j)
      if i == n-1:
        return(True)
      else:
        extendsoln = placequeen(i+1)
      if extendsoln:
        return(True)
      else:
        undoqueen(i,j)
  else:
    return(False)

board = {}
n = int(input("How many queens? "))
initialize(n)
if placequeen(0):
  printboard()
```

```
How many queens? 9
(0, 0)
(1, 2)
(2, 5)
(3, 7)
(4, 1)
(5, 3)
(6, 8)
(7, 6)
(8, 4)
```

Global vs local variables in Python

In [5]:
```
def f():
  y = x
  print(y)

x = 7
f()
```

```
7
```

In [6]:
```
def f():
  y = x
  print(y)
  x = 22

x = 7
f()
```

```
---------------------------------------------------------------------------
UnboundLocalError                         Traceback (most recent call last)
/tmp/ipykernel_353618/3077953959.py in <module>
      5
      6 x = 7
----> 7 f()

/tmp/ipykernel_353618/3077953959.py in f()
      1 def f():
----> 2   y = x
      3   print(y)
      4   x = 22
      5

UnboundLocalError: local variable 'x' referenced before assignment
```

In [7]:
```
def f():
  y = x[0]
  print("old",y)
  x[0] = 22
  y = x[0]
  print("new",y)

x = [7]
f()
```

```
old 7
new 22
```

In [8]:
```
def f():
  global x
  y = x
  print(y)
  x = 22

x = 7
f()
print(x)
```

```
7
22
```