NPTEL MOOC

PROGRAMMING, DATA STRUCTURES AND ALGORITHMS IN PYTHON

Week 6, Lecture 2

Madhavan Mukund, Chennai Mathematical Institute
http://www.cmi.ac.in/~madhavan

Recall 8 queens

```
def placequeen(i,board): # Trying row i
  for each c such that (i,c) is available:
    place queen at (i,c) and update board
    if i == n-1:
      return(True) # Last queen has been placed
    else:
      extendsoln = placequeen(i+1,board)
    if extendsoln:
      return(True) # This solution extends fully
    else:
      undo this move and update board
  else:
    return(False) # Row i failed
```

Global variables

- Can we avoid passing `board` explicitly to each function?
- Can we have a single global copy of `board` that all functions can update?

Scope of name

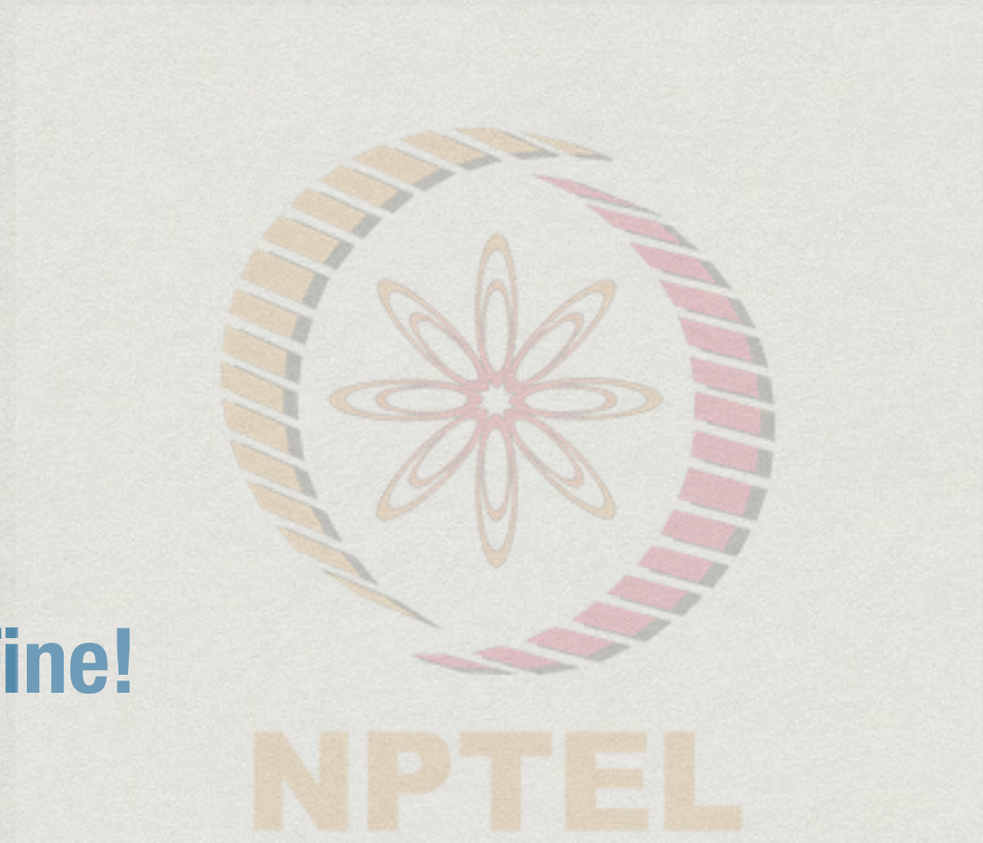
- Scope of name is the portion of code where it is available to read and update
- By default, in Python, scope is local to functions
  - But actually, only if we update the name inside the function

Two examples

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

x = 7
f()
```

Fine!



Two examples

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

x = 7
f()
```

Fine!

```
def f():
    y = x
    print(y)
    x = 22

x = 7
f()
```

Error!

NPTEL

Two examples

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

x = 7
f()
```

Fine!

```
def f():
    y = x
    print(y)
    x = 22

x = 7
f()
```

Error!

NPTEL

- If `x` is not found in `f()`, Python looks at enclosing function for global `x`
- If `x` is updated in `f()`, it becomes a local name!

Global variables

- Actually, this applies only to immutable values

- Global names that point to mutable values can be updated within a function

```
def f():
    y = x[0]
    print(y)
    x[0] = 22

x = [7]
f()
```

Fine!

NPTEL

Global immutable values

- What if we want a global integer
  - Count the number of times a function is called
- Declare a name to be `global`

```
def f():
    global x
    y = x
    print(y)
    x = 22

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

NPTEL

Global immutable values

- What if we want a global integer
  - Count the number of times a function is called
- Declare a name to be `global`

```
def f():
    global x
    y = x
    print(y)
    x = 22

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

22

NPTEL

Nest function definitions

- Can define local "helper" functions
- `g()` and `h()` are only visible to `f()`
- Cannot be called directly from outside

```
def f():
    def g(a):
        return(a+1)

    def h(b):
        return(2*b)

    global x
    y = g(x) + h(x)
    print(y)
    x = 22

x = 7
f()
```

NPTEL

Nest function definitions

- If we look up `x`, `y` inside `g()` or `h()` it will first look in `f()`, then outside
- Can also declare names global inside `g()`, `h()`
- Intermediate scope declaration: `nonlocal`
  - See Python documentation

```
def f():
    def g(a):
        return(a+1)

    def h(b):
        return(2*b)

    global x
    y = g(x) + h(x)
    print(y)
    x = 22

x = 7
f()
```

NPTEL

Summary

- Python names are looked up inside-out from within functions
- Updating a name with immutable value creates a local copy of that name
  - Can update global names with mutable values
- Use `global` definition to update immutable values
- Can nest helper function — hidden to the outside