

# §10.0-10.7, Py tut §9.0-9.2: Namespaces and Scope

---

17 Nov 2008

CMPT14x

Dr. Sean Ho

Trinity Western University

# Creating namespaces

- The **default** namespace is present as long as the Python interpreter/compiler is active
  - Contains **built-in** names like **abs()**, **float()**, **ZeroDivisionError**, etc.
- Each **module** has a **global** namespace visible everywhere in that module
  - Variables defined in the outermost level of your Python **file**
- Each **function** invocation and **class** definition also defines a new **local** namespace
  - Can be **nested**

# Example of namespaces

```
G1 = 'global'
```

```
def factorial(n):  
    L1 = 'local'  
    if n == 0 or n == 1:  
        return 1  
    return n * factorial(n-1)
```

*File module's global namespace (`__main__`)*

*Local namespace for each call to factorial*

# Scope

- “A **scope** is a **textual** region of a Python program where a namespace is **directly accessible**.”
  - Can access without using **module** name
    - ◆ e.g., **pi** rather than **math.pi**
- Scope deals with the **order** in which namespaces are searched to **resolve** a name
  - First search **local** scope
  - Then search **enclosing** functions/classes
  - Then search **global** scope for that file/module
  - Then search **built-in** names

# New names add to local scope

- New names are **created** by:
  - **Assignment**: `x = 5`
  - **Function definitions**: `def factorial(n):`
  - **Class definitions**: `class Fraction:`
  - **Imports**: `from math import *`
- **New** names always add to the **local** scope

```
def distance(x1, y1, x2, y2):  
    from math import sqrt  
    return sqrt((x2-x1)**2 + (y2-y1)**2)  
sqrt                # not defined here!
```

# The *global* directive

- Names outside the **local** scope are **read-only**
  - Attempts to **modify** them result in creating a new **local copy**

```
G1 = 'global'  
def fun():  
    G1 = 'local'      # creates local copy of G1  
fun()  
G1                  # G1 is unchanged
```

- The **global** directive says that references to those names refer to the file/module's **global** scope

# Backtracking: recursion appl.

- Knight's tour classic chess problem:
  - Find a sequence of legal **knight** moves that touches **every square** of the board once
    - ◆ Input: **size** of board, **starting** position
    - ◆ Output: sequence of board **coordinates** (x,y)
- Algorithm:
  - Find **possible** moves from current position
    - ◆ Omit squares we've already **touched**
  - For each move, take the move and **recurse**
  - If no possible moves, **return** (backtrack)