NPTEL MOOC

PROGRAMMING, DATA STRUCTURES AND ALGORITHMS IN PYTHON

Week 2, Lecture 5

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

A typical Python program

```
def function_1(..,..):
  …
def function_2(..,..):
  …
     ⋮
def function_k(..,..):
  …

statement_1
statement_2
   ⋮
statement_n
```

- Interpreter executes statements from top to bottom
- Function definitions are "digested" for future use
- Actual computation starts from `statement_1`

Function definition

```
def f(a,b,c):
  statement_1
  statement_2
  ..
  return(v)
  ..
```

- Function name, arguments/parameters
- Body is indented
- `return()` statement exits and returns a value

Passing values to functions

- Argument value is substituted for name

```
def power(x,n):
  ans = 1
  for i in range(0,n):
    ans = ans*x
  return(ans)
```

```
power(3,5)
   ↓
x = 3
n = 5
ans = 1
for i in range..
```

- Like an implicit assignment statement

Passing values …

- Same rules apply for mutable, immutable values
  - Immutable value will not be affected at calling point
  - Mutable values will be affected

Example

```
def update(l,i,v):
  if i >= 0 and i < len(l):
    l[i] = v
    return(True)
  else:
    v = v+1
    return(False)
```

```
ns = [3,11,12]
z = 8
update(ns,2,z)
update(ns,4,z)
```

- `ns` is `[3,11,8]`
- `z` remains `8`

- Return value may be ignored
- If there is no `return()`, function ends when last statement is reached

Scope of names

- Names within a function have local scope

```
def stupid(x):
  n = 17
  return(x)

n = 7
v = stupid(28)
# What is n now?
```

- n is still 7
  - Name n inside function is separate from n outside

Defining functions

- A function must be defined before it is invoked

- This is OK

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

def g(y):
  return(y+3)

z = f(77)
```

- This is not

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

z = f(77)

def g(y):
  return(y+3)
```

Recursive functions

- A function can call itself — recursion

```
def factorial(n):
  if n <= 0:
    return(1)
  else:
    val = n * factorial(n-1)
    return(val)
```

Summary

- Functions are a good way to organise code in logical chunks
- Passing arguments to a function is like assigning values to names
  - Only mutable values can be updated
- Names in functions have local scope
- Functions must be defined before use
- Recursion — a function can call itself