

NPTEL MOOC

PROGRAMMING, DATA STRUCTURES AND ALGORITHMS IN PYTHON

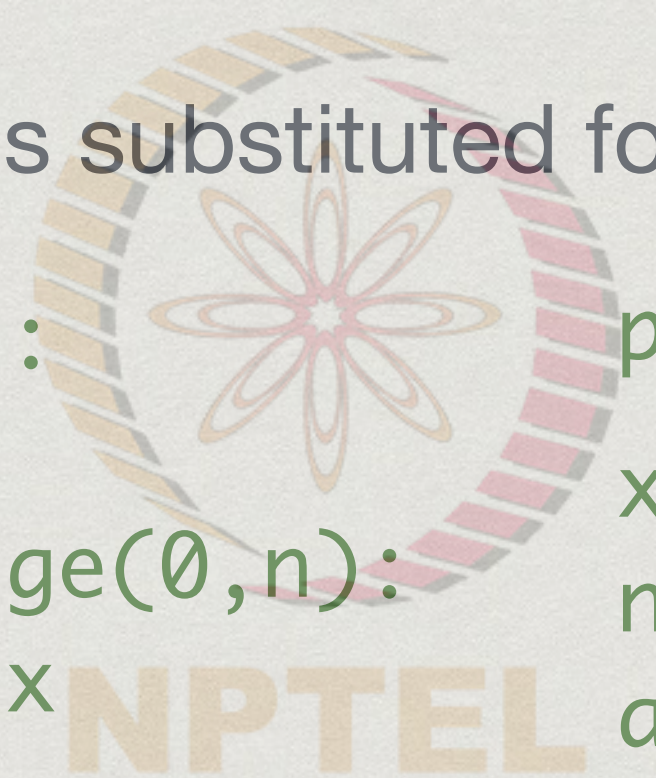
Week 4, Lecture 6

Madhavan Mukund, Chennai Mathematical Institute

<http://www.cmi.ac.in/~madhavan>

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..
```

The diagram illustrates the argument substitution process. A red arrow points from the value '3' in the function call `power(3,5)` to the parameter `x` in the function definition `def power(x,n):`. This indicates that the value 3 is substituted for the variable x within the function's scope.

- * Like an implicit assignment statement

Pass arguments by name

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

- * Call power(n=5,x=4)

Default arguments

- * Recall `int(s)` that converts string to integer
 - * `int("76")` is 76
 - * `int("A5")` generates an error
- * Actually `int(s,b)` takes two arguments, string `s` and base `b`
 - * `b` has default value 10
 - * `int("A5",16)` is 165 ($10 \times 16 + 5$)

Default arguments

```
def int(s,b=10):  
    . . .
```

- * Default value is provided in function definition
- * If parameter is omitted, default value is used
- * Default value must be available at definition time
- * `def Quicksort(A,l=0,r=len(A)):` does not work

Default arguments

```
def f(a,b,c=14,d=22):
```

```
    . . .
```

- * `f(13,12)` is interpreted as `f(13,12,14,22)`
- * `f(13,12,16)` is interpreted as `f(13,12,16,22)`
- * Default values are identified by position, must come at the end
 - * Order is important

Function definitions

- * `def` associates a function body with a name
- * Flexible, like other value assignments to name
- * Definition can be conditional

```
if condition:  
    def f(a,b,c):  
        . . .  
else:  
    def f(a,b,c):  
        . . .
```


Function definitions

- * Can assign a function to a new name

```
def f(a,b,c):
```

```
    . . .
```

```
g = f
```

- * Now `g` is another name for `f`



Can pass functions

- * Apply f to x n times

```
def apply(f,x,n):  
    res = x  
    for i in range(n):  
        res = f(res)  
    return(res)  
  
def square(x):  
    return(x*x)  
  
apply(square,5,2)  
square(square(5))
```

625

Passing functions

- * Useful for customizing functions such as sort
- * Define `cmp(x,y)` that returns `-1` if `x < y`,
`0` if `x == y` and `1` if `x > y`
- * `cmp("aab", "ab")` is `-1` in dictionary order
- * `cmp("aab", "ab")` is `1` if we compare by length
- * `def sortfunction(l, cmpfn=defaultcmpfn):`

Summary

- * Function definitions behave like other assignments of values to names
- * Can reassign a new definition, define conditionally
...
- * Can pass function names to other functions