Introduction to Computer Programming Lecture 4.3:

Recursive Functions

Hemma Philamore

Department of Engineering Mathematics

Recursive Function

Recursive functions are functions that **call themselves**.

Recursion can help solve complex problems by breaking them up **into smaller parts**.

Example:
A function pow(x, n) that raises x to a natural power (1, 2, 3 …) of n
(multiplies x by itself n times).

pow(2, 2) = 4
pow(2, 3) = 8
pow(2, 4) = 16

Recursive Function

Recursive functions are functions that **call themselves**.

Every recursion has two basic parts:

- A way to **reduce** the problem **to something smaller and similar (calls itself** to solve the subproblem)
- A **condition to stop** the recursion (often to return a trivial solution)

pow(2, 1) = 2

Stops the program from entering infinite recursion loop.

Recursive Function

Iterative

```
def pow(x, n):
    result = 1;

    // multiply result by x n times
    for i in n:
        result *= x

    return result

pow(2, 3)
```

8

Recursive

```
def pow(x, n):
    if (n == 1):
        return x

    // multiply result by x n times
    else:
        return x * pow(x, n-1)

pow(2, 3)
```

condition to end recursion

call function with a simpler version of the problem

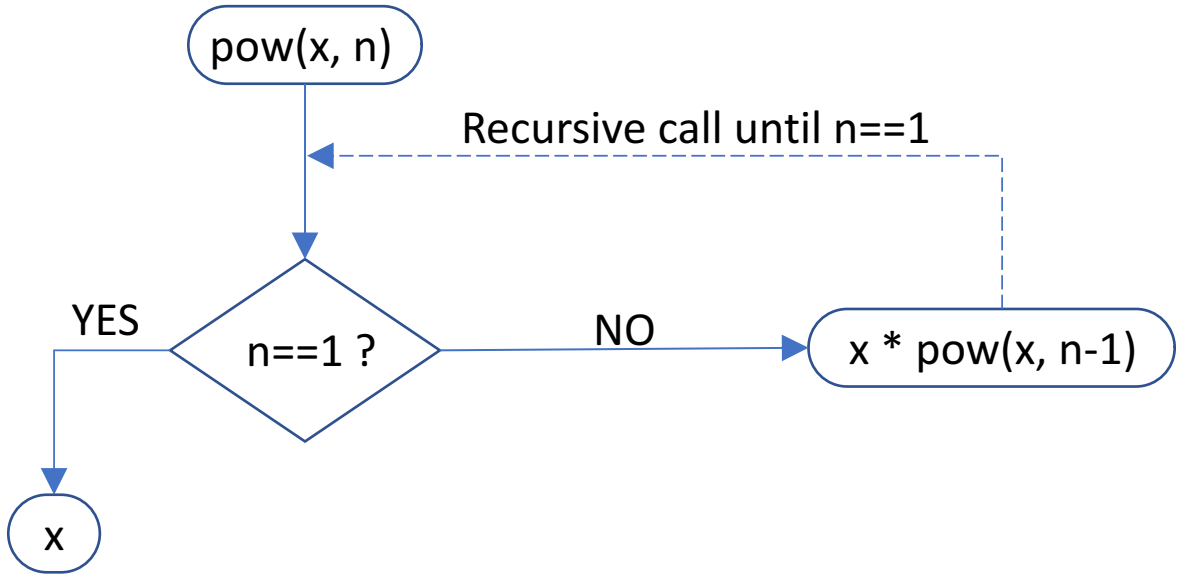
8

When pow(x, n) is called, the following control flow executes:

```
function pow(x, n):
    if (n == 1):
        return x

    // multiply result by x n times
    else:
        return x * pow(x, n-1)


pow(2, 3)
```



Recursive Function

Every recursion has two basic parts:

- A way to **reduce** the problem **to something smaller and similar (calls itself** to solve the subproblem)
- A **condition to stop** the recursion (often to return a trivial solution)

Example: Factorial

$$n! = n \cdot (n-1) \cdot (n-2) \cdot \ldots \cdot 2 \cdot 1$$

condition to end recursion

$$\ldots \cdot 2 \cdot 1$$

$n! =$

Example: Factorial

$$n! = n \cdot (n-1) \cdot (n-2) \cdot \ldots \cdot 2 \cdot 1$$

condition to end recursion

$$n! = \qquad \ldots \cdot 2 \cdot 1$$

$$0! = 1$$

Recursive Function

```
def Factorial(Num):
    if Num == 0:
        return 1
    else:
        return(Num*Factorial(Num-1))
```

condition to end recursion

call function again with a simpler version of the problem

```
>>> Factorial( 0 )
1

>>> Factorial( 1 )
1

>>> Factorial( 2 )
2
```