

1

Programming in Python¹

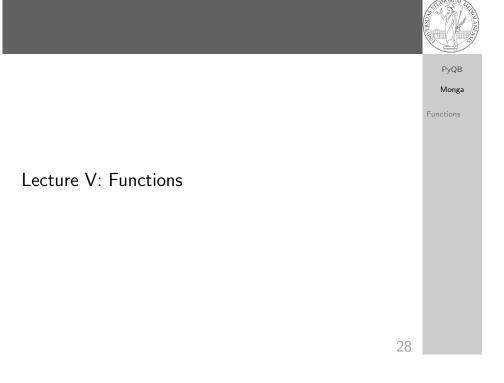
Mattia Monga

Dip. di Informatica Università degli Studi di Milano, Italia mattia.monga@unimi.it

Academic year 2021/22, II semester

1 ⊕⊕@ 2022 M. Monga. Creative Commons Attribuzione — Condividi allo stesso modo 4.0 Internazionale. http://creativecommons.org/licenses/by-sa/4.0/deed.it

A	A function computes a	a result	
	Returns a useful result		PyQB
	<pre>def concat_with_a_space(return string1 + ' '</pre>		Monga
	<pre># string1 is the _formal.</pre>	_ parameter parameter (like an assigment string1 =	Functions
	 Return None def repeated_print(strin for i in range(0, rep print(string) 	· ·	
	<pre>repeatedPrint('Hello, wo</pre>	rld!', 3)	
	<pre> • Recursive call: def repeatedPrint(string if repetitions > 0: print(string) repeatedPrint(string) </pre>	-	



Functions are objects too

Monga One can assign functions to variables: def cube(x: int) -> int: square = x * x return square * x mycube = cube print(mycube(3)) print(type(mycube)) And short functions can even be expressed as literal expressions (lambda expressions) cube = lambda y: y*y*y

repeatedPrint('Hello, world!', 3)

PyQB

The tower of Hanoi



PyQB

Monga

Functions

Describe the moves for a solution



PyQB

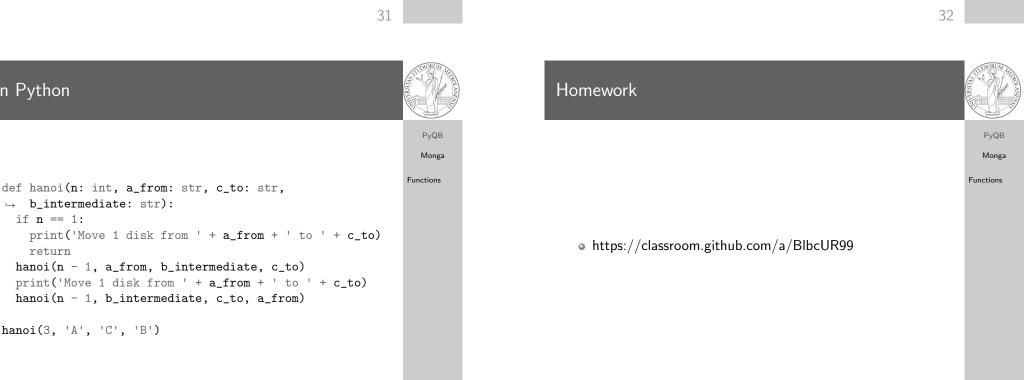
Monga

Functions

Recursive thinking is a powerful problem solving technique and it can be translated to Python thanks to recursive calls. Hanoi moves $A \rightarrow C$:

- In A there is just one disk: move it to C
- Otherwise in A there are n disks (> 1):
 - leap of faith! I suppose to know the moves needed to move n-1 disk; then
 - apply this (supposed) solution to move n-1 disks from A to B (leveraging on C, empty, as the third pole)
 - move the last disk from A to C
 - apply the (supposed) solution to move n-1 disks from B to C (leveraging on A, now empty, as the third pole)

This implicit description solve the problem! Finding a non-recursive solution is possible but not that easy.



https://www.mathsisfun.com/games/towerofhanoi.html

In Python

 \rightarrow b_intermediate: str): if n == 1: print('Move 1 disk from ' + a_from + ' to ' + c_to) return hanoi(n - 1, a_from, b_intermediate, c_to) print('Move 1 disk from ' + a_from + ' to ' + c_to) hanoi(n - 1, b_intermediate, c_to, a_from) hanoi(3, 'A', 'C', 'B')

33