Programming in Python[1]

Mattia Monga

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

Academic year 2021/22, II semester

Lecture V: Functions

A function computes a result

- Returns a useful result

```
def concat_with_a_space(string1, string2):
   return string1 + ' ' + string2

# string1 is the _formal_ parameter
# 'foo' is the _actual_ parameter (like an assigment string1 =
↪  'foo')
print(concat_with_a_space('foo','bar'))
```

- Return None

```
def repeated_print(string, repetitions):
   for i in range(0, repetitions):
     print(string)

repeatedPrint('Hello, world!', 3)
```

- **Recursive call**:

```
def repeatedPrint(string, repetitions):
   if repetitions > 0:
     print(string)
     repeatedPrint(string, repetitions - 1)

repeatedPrint('Hello, world!', 3)
```

Functions are objects too

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
```

Naming helps solving

The tower of Hanoi
https://www.mathsisfun.com/games/towerofhanoi.html

Describe the moves for a solution

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.

In Python

```python
def hanoi(n: int, a_from: str, c_to: str,
↪  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')
```

Homework

- https://classroom.github.com/a/BIbcUR99