

Corso di *STATISTICA, INFORMATICA, ELABORAZIONE DELLE INFORMAZIONI*

Modulo di Sistemi di Elaborazione delle Informazioni

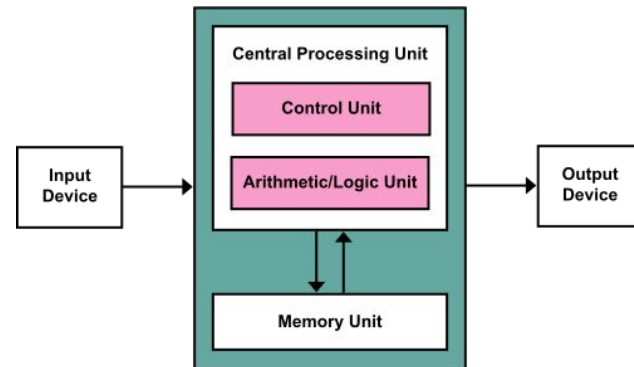
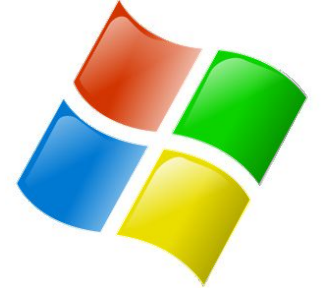


UNIVERSITÀ DEGLI STUDI DELLA BASILICATA



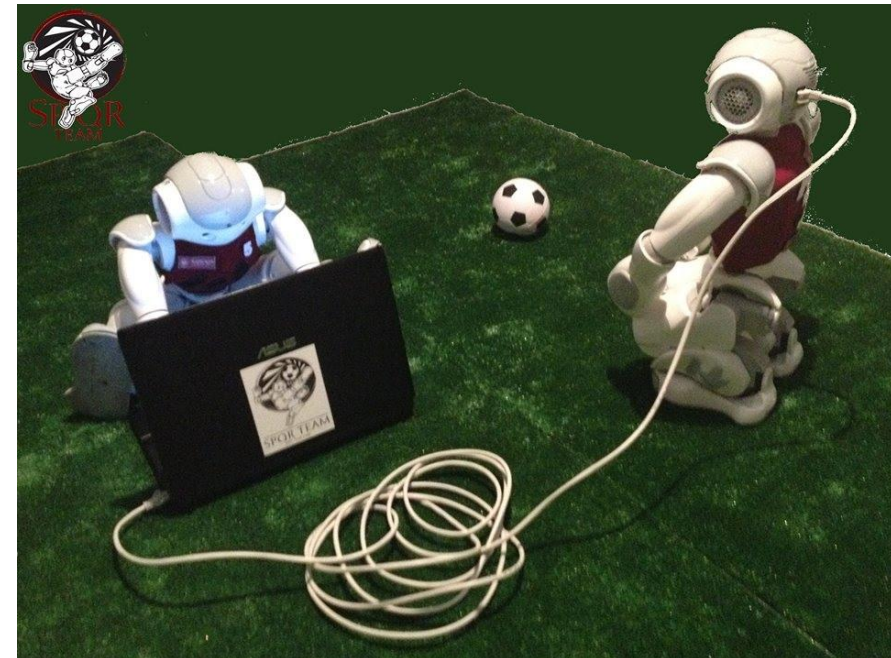
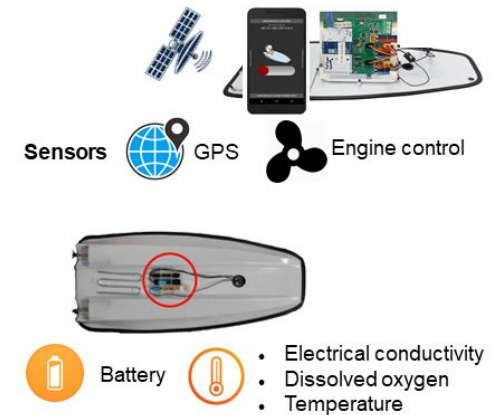
Funzioni

Docente:
Domenico Daniele Bloisi



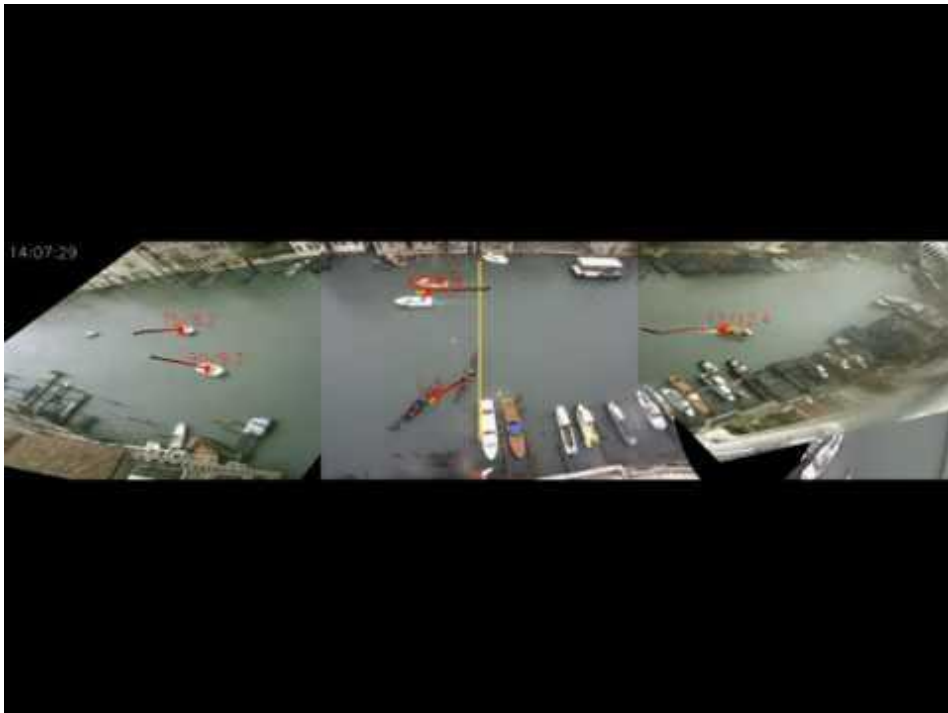
Domenico Daniele Bloisi

- Professore Associato
Dipartimento di Matematica, Informatica
ed Economia
Università degli studi della Basilicata
<http://web.unibas.it/bloisi>
- SPQR Robot Soccer Team
Dipartimento di Informatica, Automatica
e Gestionale Università degli studi di
Roma “La Sapienza”
<http://spqr.diag.uniroma1.it>



Interessi di ricerca

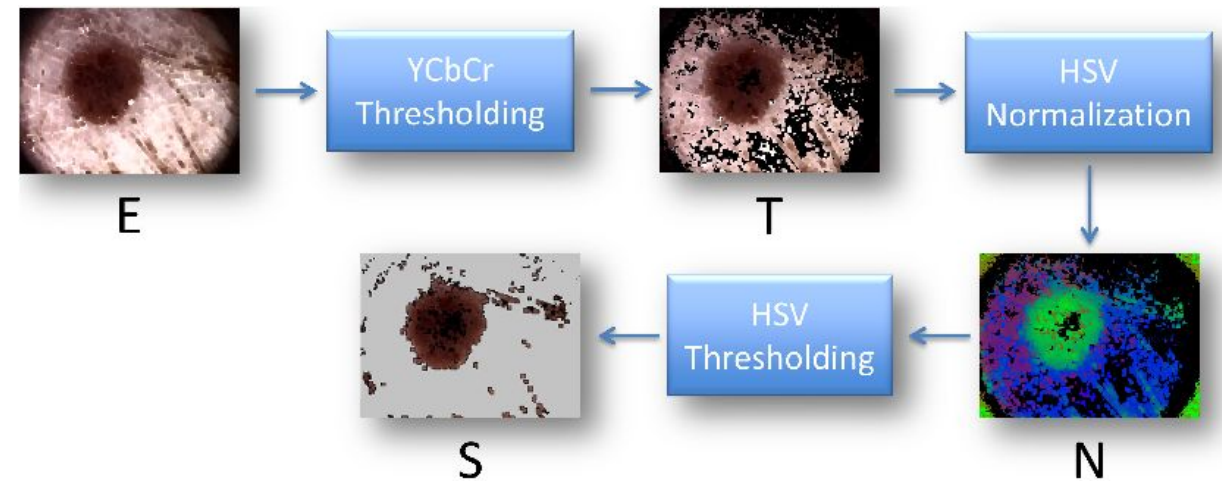
- Intelligent surveillance
- Robot vision
- Medical image analysis



https://youtu.be/9a70Ucgbi_U



<https://youtu.be/2KHNZX7UIWQ>



UNIBAS Wolves <https://sites.google.com/unibas.it/wolves>



- UNIBAS WOLVES is the robot soccer team of the University of Basilicata. Established in 2019, it is focussed on developing software for NAO soccer robots participating in RoboCup competitions.

- UNIBAS WOLVES team is twinned with SPQR Team at Sapienza University of Rome



<https://youtu.be/ji0OmkaWh20>

Informazioni sul corso

Il corso di STATISTICA, INFORMATICA, ELABORAZIONE DELLE INFORMAZIONI

- include 3 moduli:
 - SISTEMI DI ELABORAZIONE DELLE INFORMAZIONI
(il martedì - docente: Domenico Bloisi)
 - INFORMATICA
(il mercoledì - docente: Enzo Veltri)
 - PROBABILITA' E STATISTICA MATEMATICA
(il giovedì - docente: Antonella Iuliano)
- Periodo: **I semestre** ottobre 2022 – gennaio 2023

Informazioni sul modulo

- Home page del modulo:
<https://web.unibas.it/bloisi/corsi/sei.html>
- Martedì dalle 11:30 alle 13:30

Ricevimento Bloisi

- In presenza, durante il periodo delle lezioni:
Lunedì dalle 17:00 alle 18:00
presso Edificio 3D, Il piano, stanza 15
Si invitano gli studenti a controllare regolarmente la bacheca degli avvisi per eventuali variazioni
- Tramite google Meet e al di fuori del periodo delle lezioni:
da concordare con il docente tramite email

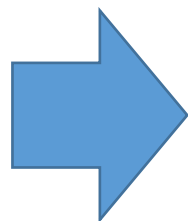
Per prenotare un appuntamento inviare
una email a
domenico.bloisi@unibas.it



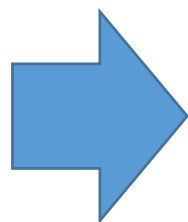
Recap

Esercizio 9

Si scriva un codice che riceva come input da tastiera due interi a e b e disegni sullo schermo un rettangolo di dimensioni $a \times b$ usando il carattere ' * ', così come mostrato negli esempi



```
Lato a: 5
Lato b: 7
* * * * *
*           *
*           *
*           *
*           *
*           *
* * * * *
```



```
Lato a: 8
Lato b: 3
* * * * * * * *
*           *
* * * * * * * *
```

Ciclo for

```
▶ a = int(input("Lato a: ")) #colonne  
b = int(input("Lato b: ")) #righe  
  
colonne = a  
righe = b  
  
for r in range(righe):  
    print("* ")
```

```
↳ Lato a: 5  
Lato b: 7  
*  
*  
*  
*  
*  
*  
*
```


Cicli for annidati

```
▶ for r in range(righe):  
    for c in range(colonne):  
        print("* ",end="")
```

```
↳ * * * * *  
* * * * *  
* * * * *  
* * * * *  
* * * * *
```

Cicli for annidati

```
▶ for r in range(righe):  
    for c in range(colonne):  
        print("* ",end="")  
        print("\n")
```



```
* * * * *
```

```
* * * * *
```

```
* * * * *
```

```
* * * * *
```

```
* * * * *
```

```
* * * * *
```

```
* * * * *
```

Possibile soluzione Esercizio 9

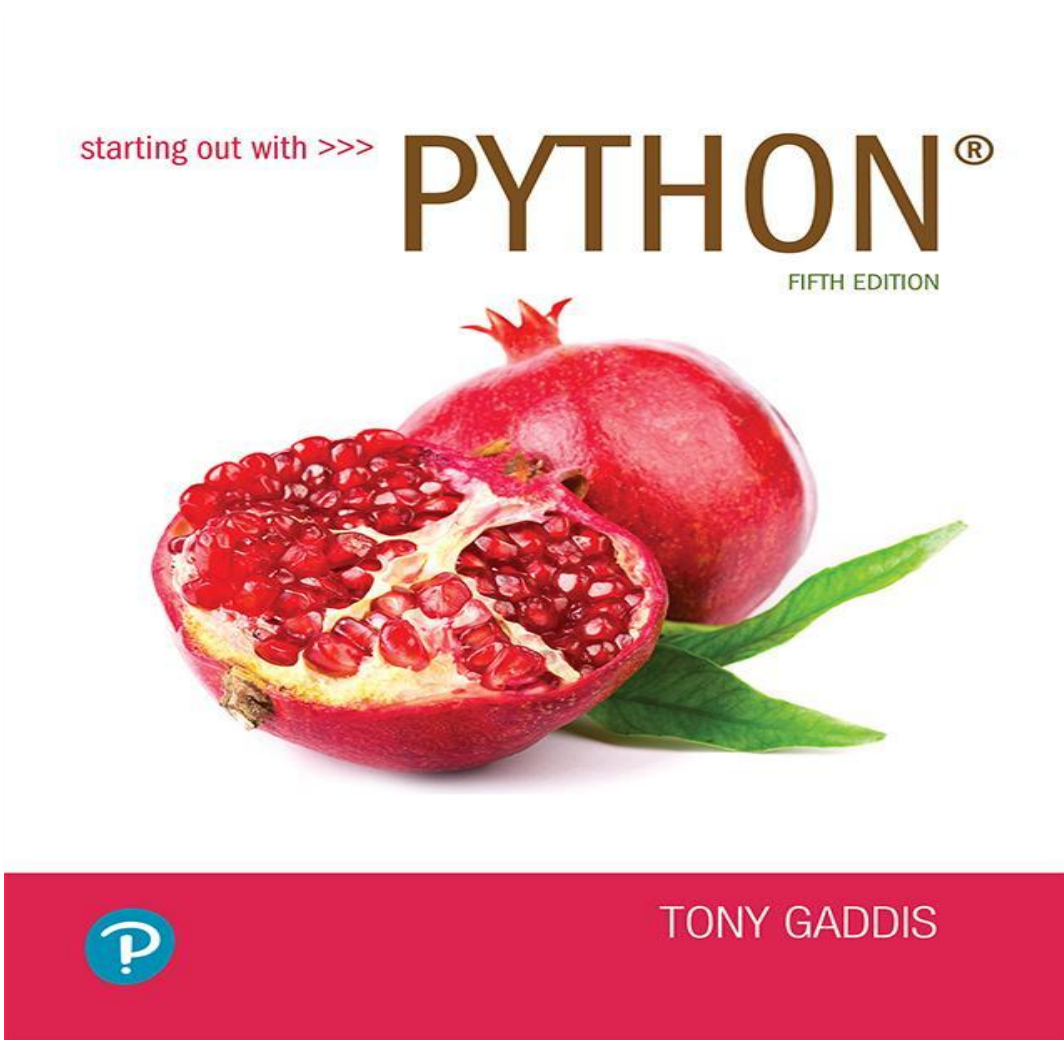
```
▶ for r in range(righe):  
    if r == 0 or r == (righe - 1):  
        for c in range(colonne):  
            print("* ",end="")  
    else:  
        for c in range(colonne):  
            if c > 0 and c < (colonne - 1):  
                print("  ",end="")  
            else:  
                print("* ",end="")  
    print("\n")
```



```
* * * * *  
  
*           *  
  
*           *  
  
*           *  
  
*           *  
  
*           *  
  
* * * * *
```

Starting out with Python

Fifth Edition



Chapter 5

Functions

Topics (1 of 2)

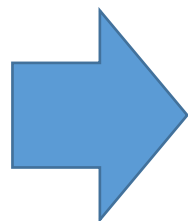
- Introduction to Functions
- Defining and Calling a Void Function
- Designing a Program to Use Functions
- Local Variables
- Passing Arguments to Functions
- Global Variables and Global Constants
- Turtle Graphics: Modularizing Code with Functions

Topics (2 of 2)

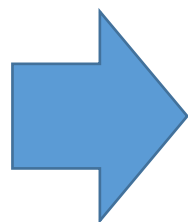
- Introduction to Value-Returning Functions: Generating Random Numbers
- Writing Your Own Value-Returning Functions
- The `math` Module
- Storing Functions in Modules

Esercizio 10

Si scriva **una funzione** riceva come input due interi a e b e disegni sullo schermo un rettangolo di dimensioni $a \times b$ usando il carattere '*', così come mostrato negli esempi



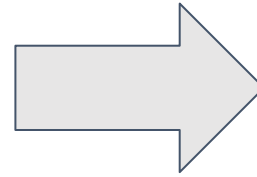
```
Lato a: 5
Lato b: 7
* * * * *
*       *
*       *
*       *
*       *
*       *
*       *
* * * * *
```



```
Lato a: 8
Lato b: 3
* * * * * * * *
*               *
* * * * * * * *
```

Soluzione Esercizio 10

```
▶ def disegna_rettangolo(righe, colonne):  
    for r in range(righe):  
        if r == 0 or r == (righe - 1):  
            for c in range(colonne):  
                print("* ",end="")  
        else:  
            for c in range(colonne):  
                if c > 0 and c < (colonne - 1):  
                    print(" ",end="")  
                else:  
                    print("* ",end="")  
    print("\n")  
  
a = int(input("Lato a: ")) #colonne  
b = int(input("Lato b: ")) #righe  
  
disegna_rettangolo(b, a)
```



```
↳ Lato a: 5  
Lato b: 7  
* * * * *  
  
*           *  
  
*           *  
  
*           *  
  
*           *  
  
*           *  
  
* * * * *
```

Introduction to Functions (1 of 2)

- Function: group of statements within a program that perform as specific task
 - Usually one task of a large program
 - Functions can be executed in order to perform overall program task
 - Known as *divide and conquer* approach
- Modularized program: program wherein each task within the program is in its own function

Introduction to Functions (2 of 2)

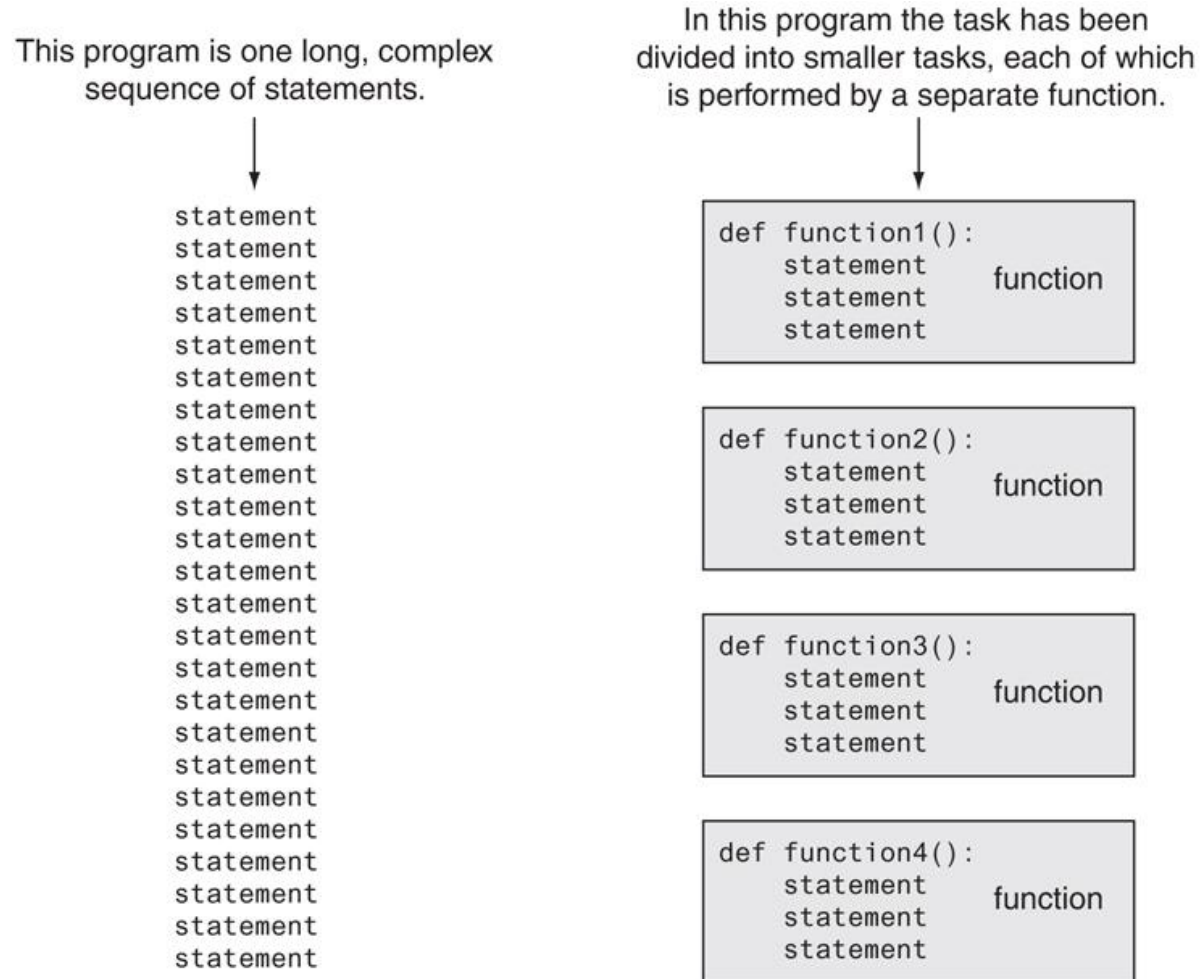


Figure 5-1 Using functions to divide and conquer a large task

Benefits of Modularizing a Program with Functions

- The benefits of using functions include:
 - Simpler code
 - Code reuse
 - write the code once and call it multiple times
 - Better testing and debugging
 - Can test and debug each function individually
 - Faster development
 - Easier facilitation of teamwork
 - Different team members can write different functions

Definizione di nuove funzioni

La definizione di una nuova funzione è composta dai seguenti elementi:

- il nome della funzione
- il numero dei suoi argomenti
- la sequenza di istruzioni, detta corpo della funzione, che dovranno essere eseguite quando la funzione sarà chiamata

La definizione di una nuova funzione avviene attraverso l'uso della keyword `def`

def

Sintassi:

```
def nome_funzione (par1, ..., parn) :  
    corpo_della_funzione
```

- `nome_funzione` è un nome simbolico scelto dal programmatore, con gli stessi vincoli a cui sono soggetti i nomi delle variabili
- `par1, ..., parn` sono nomi (scelti dal programmatore) di variabili, dette parametri della funzione, alle quali l'interprete assegnerà i valori degli argomenti che verranno indicati nella chiamata della funzione
- `corpo_della_funzione` è una sequenza di una o più istruzioni qualsiasi, ciascuna scritta in una riga distinta, con un rientro di almeno un carattere, identico per tutte le istruzioni

La prima riga della definizione (contenente i nomi della funzione e dei parametri) è detta intestazione della funzione.

Void Functions and Value-Returning Functions

- A void function:
 - Simply executes the statements it contains and then terminates.
- A value-returning function:
 - Executes the statements it contains, and then it returns a value back to the statement that called it.
 - The `input`, `int`, and `float` functions are examples of value-returning functions.

return

Per concludere l'esecuzione di una funzione e indicare il valore che la funzione dovrà restituire come risultato della sua chiamata si usa l'istruzione `return`.

Sintassi:

```
return espressione
```

dove `espressione` è un'espressione Python qualsiasi.

L'istruzione `return` può essere usata solo solo all'interno di una funzione.

Se una funzione non deve restituire alcun valore:

- l'istruzione `return` può essere usata, senza l'indicazione di alcuna espressione, per concludere l'esecuzione della funzione
- se non si usa l'istruzione `return`, l'esecuzione della funzione terminerà dopo l'esecuzione dell'ultima istruzione del corpo

Definizione e chiamata di una funzione

L'esecuzione dell'istruzione `def` non comporta l'esecuzione delle istruzioni della funzione: tali istruzioni verranno eseguite solo attraverso una chiamata della funzione.

L'istruzione `def` dovrà essere eseguita una sola volta, prima di qualsiasi chiamata della funzione. In caso contrario, il nome della funzione non sarà riconosciuto dall'interprete e la chiamata produrrà un messaggio di errore.

Esecuzione della chiamata di funzione

L'interprete esegue la chiamata di una funzione nel modo seguente:

1. copia il valore di ciascun argomento nel parametro corrispondente (quindi tali variabili possiedono già un valore nel momento in cui inizia l'esecuzione della funzione)
2. esegue le istruzioni del corpo della funzione, fino all'istruzione `return` oppure fino all'ultima istruzione del corpo
3. se l'eventuale istruzione `return` è seguita da un'espressione, restituisce il valore di tale espressione come risultato della chiamata

Defining and Calling a Function (1 of 5)

- Functions are given names
 - Function naming rules:
 - Cannot use keywords as a function name
 - Cannot contain spaces
 - First character must be a letter or underscore
 - All other characters must be a letter, number or underscore
 - Uppercase and lowercase characters are distinct

Defining and Calling a Function (2 of 5)

- Function name should be descriptive of the task carried out by the function
 - Often includes a verb
- Function definition: specifies what function does

```
def function_name():  
    statement  
    statement
```

Defining and Calling a Function (2 of 5)



```
def stampa_di_prova():  
    print("asdfghjkl")  
    print(1234567)
```

Defining and Calling a Function (3 of 5)

- Function header: first line of function
 - Includes keyword `def` and function name, followed by parentheses and colon
- Block: set of statements that belong together as a group
 - Example: the statements included in a function

Defining and Calling a Function (3 of 5)

```
def stampa_di_prova():  
    print("asdfghjkl")  
    print(1234567)
```

Header

Block

Defining and Calling a Function (4 of 5)

- Call a function to execute it
 - When a function is called:
 - Interpreter jumps to the function and executes statements in the block
 - Interpreter jumps back to part of program that called the function
 - Known as function return

Defining and Calling a Function (4 of 5)

```
def stampa_di_prova():  
    print("asdfghjkl")  
    print(1234567)  
  
stampa_di_prova()
```

Header

Block

Chiamata

```
asdfghjkl  
1234567
```

Defining and Calling a Function (5 of 5)

- main function: called when the program starts
 - Calls other functions when they are needed
 - Defines the *mainline logic* of the program

Defining and Calling a Function (5 of 5)

```
▶ def main():  
    print("punto di ingresso del programma")  
    stampa_di_prova()  
    print("arrivederci")  
  
def stampa_di_prova():  
    print("asdfghjkl")  
    print(1234567)  
  
main()
```

```
punto di ingresso del programma  
asdfghjkl  
1234567  
arrivederci
```

Indentation in Python

- Each block must be indented
 - Lines in block must begin with the same number of spaces
 - Use tabs or spaces to indent lines in a block, but not both as this can confuse the Python interpreter
 - IDLE automatically indents the lines in a block
 - Blank lines that appear in a block are ignored

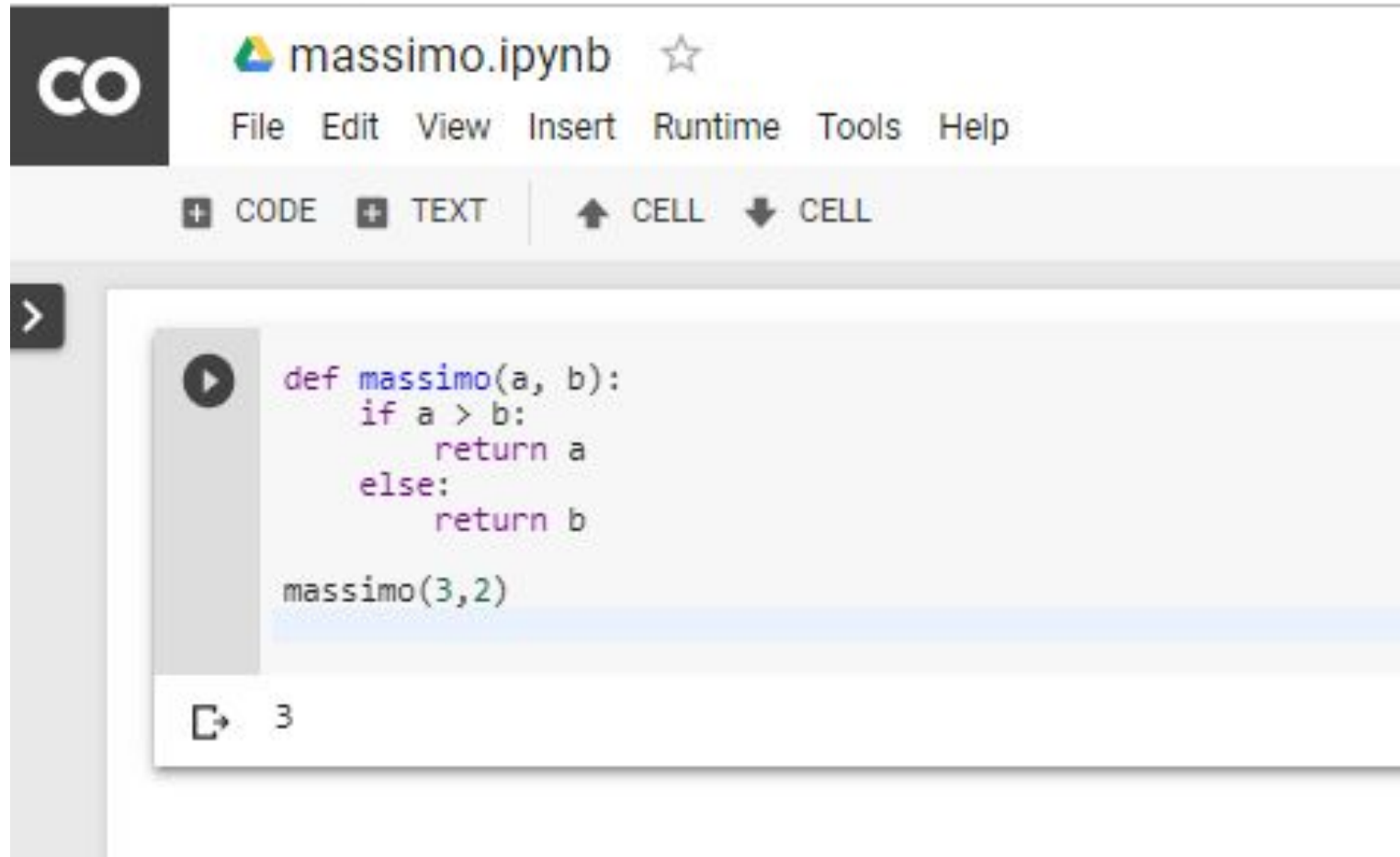
Definizione di funzioni: esempio

Si supponga di voler definire una funzione che restituisca il più grande tra due numeri ricevuti come argomenti.

Scegliendo `massimo` come nome della funzione, e `a` e `b` come nomi dei suoi parametri, la funzione può essere definita come segue:

```
def massimo(a, b):  
    if a > b:  
        return a  
    else:  
        return b
```

Esecuzione di funzione



The screenshot shows a Jupyter Notebook interface. At the top left is the 'CO' logo. The notebook title is 'massimo.ipynb' with a star icon. Below the title is a menu bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', and 'Help'. A toolbar contains '+ CODE', '+ TEXT', '↑ CELL', and '↓ CELL'. The main area shows a code cell with a play button icon on the left. The code defines a function 'massimo(a, b)' that returns 'a' if 'a > b' and 'b' otherwise, followed by a call to 'massimo(3,2)'. Below the code, a small box shows the output '3' with a copy icon.

```
def massimo(a, b):  
    if a > b:  
        return a  
    else:  
        return b  
  
massimo(3,2)
```

3

Corso di *STATISTICA, INFORMATICA, ELABORAZIONE DELLE INFORMAZIONI*

Modulo di Sistemi di Elaborazione delle Informazioni



UNIVERSITÀ DEGLI STUDI DELLA BASILICATA



Funzioni

Docente:
Domenico Daniele Bloisi

