

Corso di **STATISTICA, INFORMATICA, ELABORAZIONE DELLE INFORMAZIONI**

Modulo di Sistemi di Elaborazione delle Informazioni

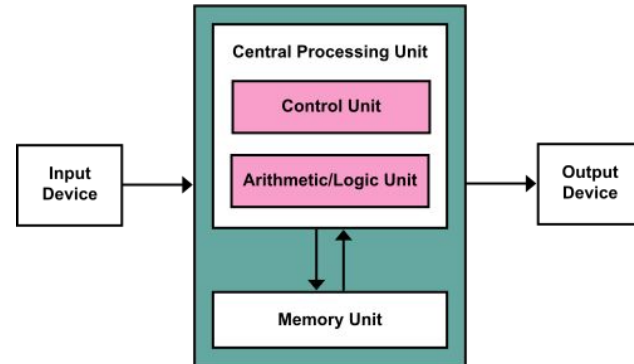
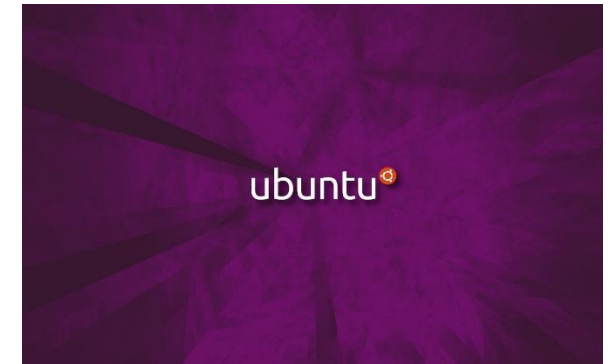
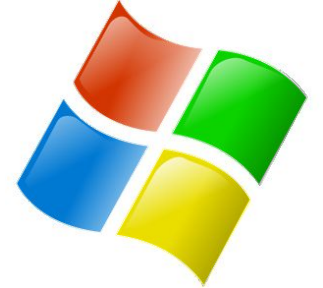
Introduzione alla programmazione



UNIVERSITÀ DEGLI STUDI DELLA BASILICATA

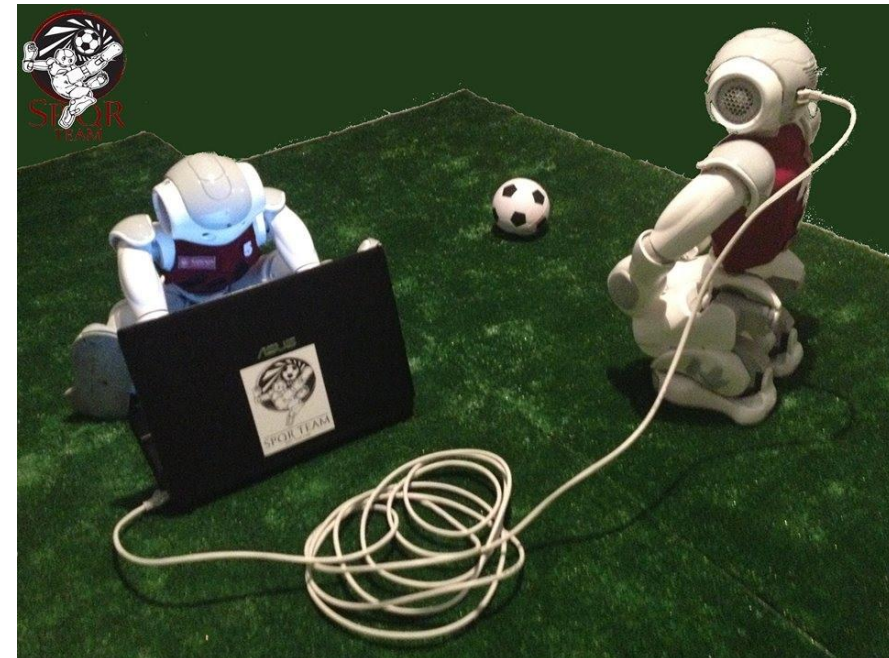
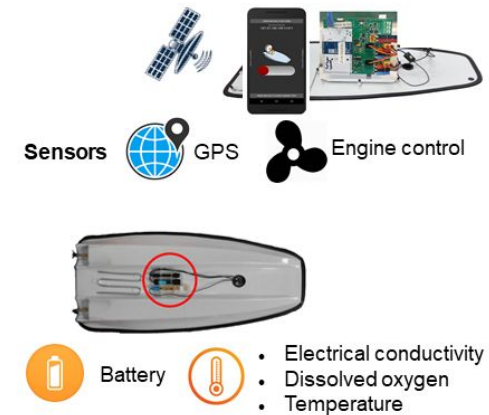


Docente:
Domenico Daniele Bloisi



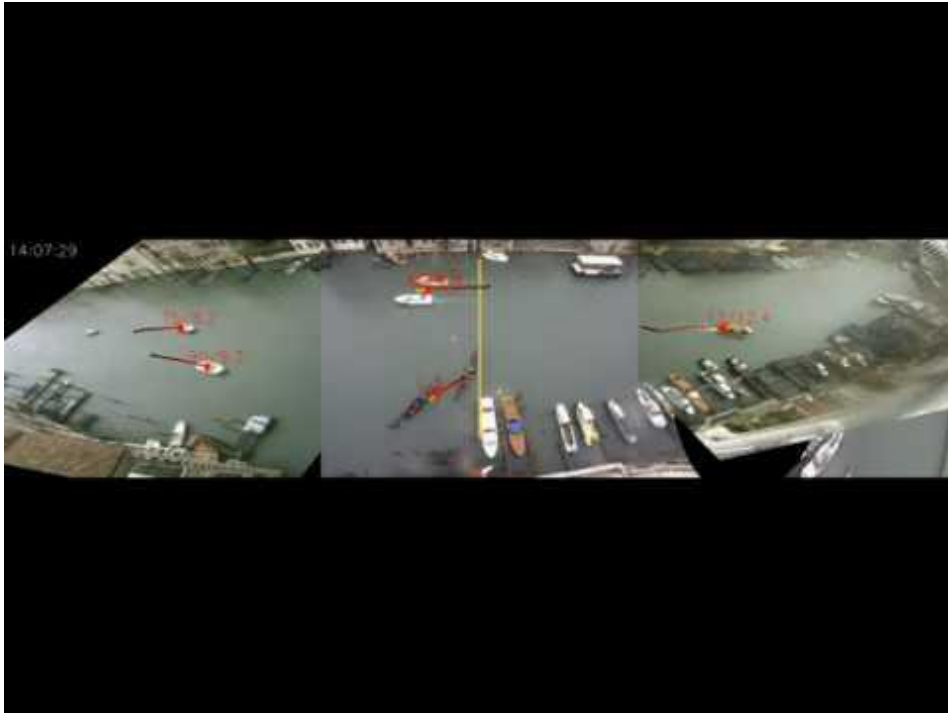
Domenico Daniele Bloisi

- 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

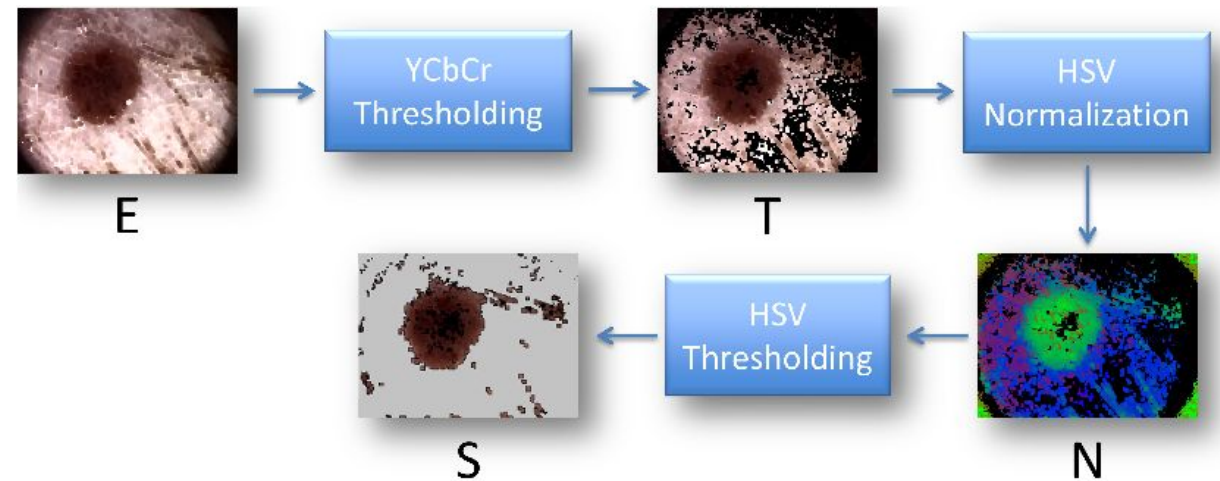
- Intelligent surveillance
- Robot vision
- Medical image analysis



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
 - Martedì dalle 11:30 alle 13:30

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



Progettazione e Sviluppo del Software

INTRODUZIONE A PYTHON - TONY GADDIS

 Guida

Le mie classi:

✓ **Sistemi di Elaborazione delle Informazioni - Domenico Bloisi (Codice classe: EKLQV7N6 )**

Comunica ai tuoi studenti il codice per iscriversi alla classe.

Crea nuova classe

Introduction

- Computers can be programmed
 - Designed to do any job that a program tells them to
- Program: set of instructions that a computer follows to perform a task
 - Commonly referred to as *Software*
- Programmer: person who can design, create, and test computer programs
 - Also known as software developer

Attori coinvolti
nello sviluppo
software

Contesto Organizzativo

- Attori coinvolti:
 - **Committente**
 - Ha una necessità informatica da risolvere
 - **Esperti del domino**
 - Conoscono il problema e sanno di cosa c'è bisogno
 - **Analista**
 - Dichiarano **cosa** serve per risolvere il problema
 - **Progettista**
 - Dichiarano **come** risolvere il problema
 - **Programmatore**
 - **Costruisce** il sistema (hardware o software)
 - **Utente finale**
 - **Utilizzatore** del sistema
 - **Manutentore**
 - **Manutiene** il sistema
-

Esempio: applicazione per gestire le multe

- Il Comune di Bugliano intende rendere automatica la gestione delle informazioni relative alle contravvenzioni sul suo territorio.
- In particolare, intende dotare ogni vigile di un dispositivo palmare che consenta di comunicare al sistema informatico
 - il veicolo a cui è stata comminata la contravvenzione
 - il luogo in cui è stata elevata tale contravvenzione
 - la natura dell'infrazione



Esempio: applicazione per gestire le multe

- Il sistema informatico provvederà a notificare, tramite posta ordinaria, la contravvenzione al cittadino interessato.
- Il Comune bandisce una gara per la realizzazione e manutenzione del sistema, che viene vinta dalla ditta "R&D spa"

Quali sono gli attori coinvolti in questa applicazione?

Esercizio: Attori coinvolti

- o Committente
 - o Esperto del dominio
 - o Utenti finali
 - o Analista
 - o Progettista
 - o Programmatore
 - o Manutentore
-

Attori coinvolti nell'esempio di riferimento

- o Committente: **Comune di Bugliano**
 - o Esperto del dominio: **vigili, funzionari del Comune o altro professionista designato, esperto del Codice della Strada**
 - o Utenti finali: **vigili e cittadini**
 - o Analista
 - o Progettista
 - o Programmatore
 - o Manutentore
- Personale della ditta "R&D spa"**

HW e SW

Hardware and Software

- Hardware: The physical devices that make up a computer
 - Computer is a system composed of several components that all work together
- Typical major components:
 - Central processing unit
 - Main memory
 - Secondary storage devices
 - Input and output devices

The CPU

- Central processing unit (CPU): the part of the computer that actually runs programs
 - Most important component
 - Without it, cannot run software
 - Used to be a huge device
- Microprocessors: CPUs located on small chips

Main Memory

- Main memory: where computer stores a program while program is running, and data used by the program
- Known as *Random Access Memory* or *RAM*
 - CPU is able to quickly access data in RAM
 - Volatile memory used for temporary storage while program is running
 - Contents are erased when computer is off

Secondary Storage Devices

- Secondary storage: can hold data for long periods of time
 - Programs normally stored here and loaded to main memory when needed
- Types of secondary memory
 - Disk drive: magnetically encodes data onto a spinning circular disk
 - Solid state drive: faster than disk drive, no moving parts, stores data in solid state memory
 - Flash memory: portable, no physical disk

Input Devices

- Input: data the computer collects from people and other devices
- Input device: component that collects the data
 - Examples: keyboard, mouse, touchscreen, scanner, camera
 - Disk drives can be considered input devices because they load programs into the main memory

Output Devices

- Output: data produced by the computer for other people or devices
 - Can be text, image, audio, or bit stream
- Output device: formats and presents output
 - Examples: video display, printer
 - Disk drives and USB drives can be considered output devices because data is sent to them to be saved

Software (1 of 2)

- Everything the computer does is controlled by software
 - General categories:
 - Application software
 - System software
- Application software: programs that make computer useful for every day tasks
 - Examples: word processing, email, games, and Web browsers

Software (2 of 2)

- System software: programs that control and manage basic operations of a computer
 - Operating system: controls operations of hardware components
 - Utility Program: performs specific task to enhance computer operation or safeguard data
 - Software development tools: used to create, modify, and test software programs

Applicazioni Software

Classificazione delle applicazioni

Le applicazioni software possono essere classificate

- **Flusso di controllo**

- Rispetto al program counter del programma

- **Elementi di interesse primario**

- Rispetto all'interesse primario di una applicazione

Ad esempio, una data funzionalità o la gestione dei dati (basi di dati) o al controllo

Classificazione rispetto al flusso di controllo

- **Sequenziali**: un unico flusso di controllo governa l'evoluzione dell'applicazione
 - **Concorrenti**: le varie attività necessitano di sincronizzazione e comunicazione
 - composte da varie attività sequenziali che possono (e devono) essere sincronizzate al fine di garantire la correttezza
 - il tempo di esecuzione influenza le prestazioni, non la correttezza
 - **Dipendenti dal tempo**: esistono vincoli temporali riguardanti sia la velocità di esecuzione delle attività sia la necessità di sincronizzare le attività stesse (c'è un riferimento temporale esterno)
-

Esempi di applicazioni legate al flusso di controllo

- **Sequenziali:**

- risolutore di sistemi di equazioni
- SO anni 80

- **Concorrenti:**

- sistema di prenotazione per camere d'albergo
- sistema operativo
- smartphone

- **Dipendenti dal tempo:**

- sistema di navigazione autonoma di un aereo
 - video gioco
-

Classificazione rispetto all'interesse primario

- **Orientate alla realizzazione di funzioni**: la complessità dominante del sistema riguarda le funzioni da realizzare
 - **Orientate alla gestione dei dati**: l'aspetto prevalente è rappresentato dai dati che vengono memorizzati, ricercati, e modificati, e che costituiscono il patrimonio informativo di una organizzazione
 - **Orientate al controllo**: la complessità prevalente del sistema riguarda il controllo delle attività che si sincronizzano e cooperano durante l'evoluzione del sistema
-

Esempi di applicazioni legate all'interesse primario

- **Orientate alla realizzazione di funzioni:**

- Matlab

- **Orientate alla gestione dei dati:**

- DBMS

- **Orientate al controllo:**

- Sistemi operativi per robot (ROS), sistemi ABS

Applicazioni più comuni

- Sequenziali e concorrenti
 - Orientate alla realizzazione di funzioni
-

Applicazioni web

Nelle applicazioni web troviamo una netta differenziazione tra il

- **Front-end**

cioè la parte visibile dagli utenti e quella con cui essi interagiscono

- **Back-end**

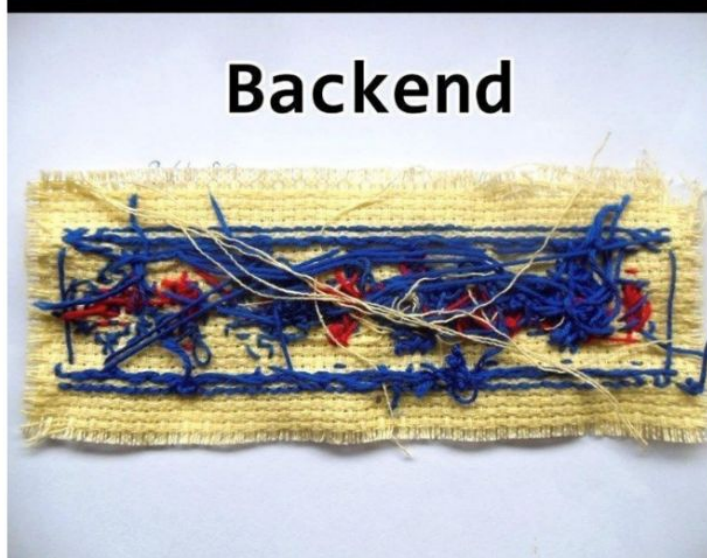
la parte riguardante il funzionamento lato server e, più in generale, ciò che gli utenti non vedono visitando il sito

Front-end vs Back-end

Frontend



Backend



Front-end

Back-end





***Backend**

***Frontend**

FULL STACK DEVELOPER

What companies think they get

Buy 1 get 1
FREE



Front-end
dev

Back-end
dev

What they actually get



Know-a-bit-of-everything
dev

Memorizzazione dei dati

How Computers Store Data

- All data in a computer is stored in sequences of 0s and 1s
- Byte: just enough memory to store letter or small number
 - Divided into eight bits
 - Bit: electrical component that can hold positive or negative charge, like on/off switch
 - The on/off pattern of bits in a byte represents data stored in the byte

Storing Numbers

- Bit represents two values, 0 and 1
- Computers use binary numbering system
 - Position of digit j is assigned the value 2^{j-1}
 - To determine value of binary number sum position values of the 1s
- Byte size limits are 0 and 255
 - 0 = all bits off; 255 = all bits on
 - To store larger number, use several bytes

Storing Characters

- Data stored in computer must be stored as binary number
- Characters are converted to numeric code, numeric code stored in memory
 - Most important coding scheme is ASCII
 - ASCII is limited: defines codes for only 128 characters
 - Unicode coding scheme becoming standard
 - Compatible with ASCII
 - Can represent characters for other languages

Advanced Number Storage

- To store negative numbers and real numbers, computers use binary numbering and encoding schemes
 - Negative numbers encoded using two's complement
 - Real numbers encoded using floating-point notation

Other Types of Data

- Digital: describes any device that stores data as binary numbers
- Digital images are composed of pixels
 - To store images, each pixel is converted to a binary number representing the pixel's color
- Digital music is composed of sections called samples
 - To store music, each sample is converted to a binary number

Funzionamento di un programma

How a Program Works (1 of 3)

- CPU designed to perform simple operations on pieces of data
 - Examples: reading data, adding, subtracting, multiplying, and dividing numbers
 - Understands instructions written in machine language and included in its instruction set
 - Each brand of CPU has its own instruction set
- To carry out meaningful calculation, CPU must perform many operations

How a Program Works (2 of 3)

- Program must be copied from secondary memory to RAM each time CPU executes it
- CPU executes program in cycle:
 - Fetch: read the next instruction from memory into CPU
 - Decode: CPU decodes fetched instruction to determine which operation to perform
 - Execute: perform the operation

How a Program Works (3 of 3)

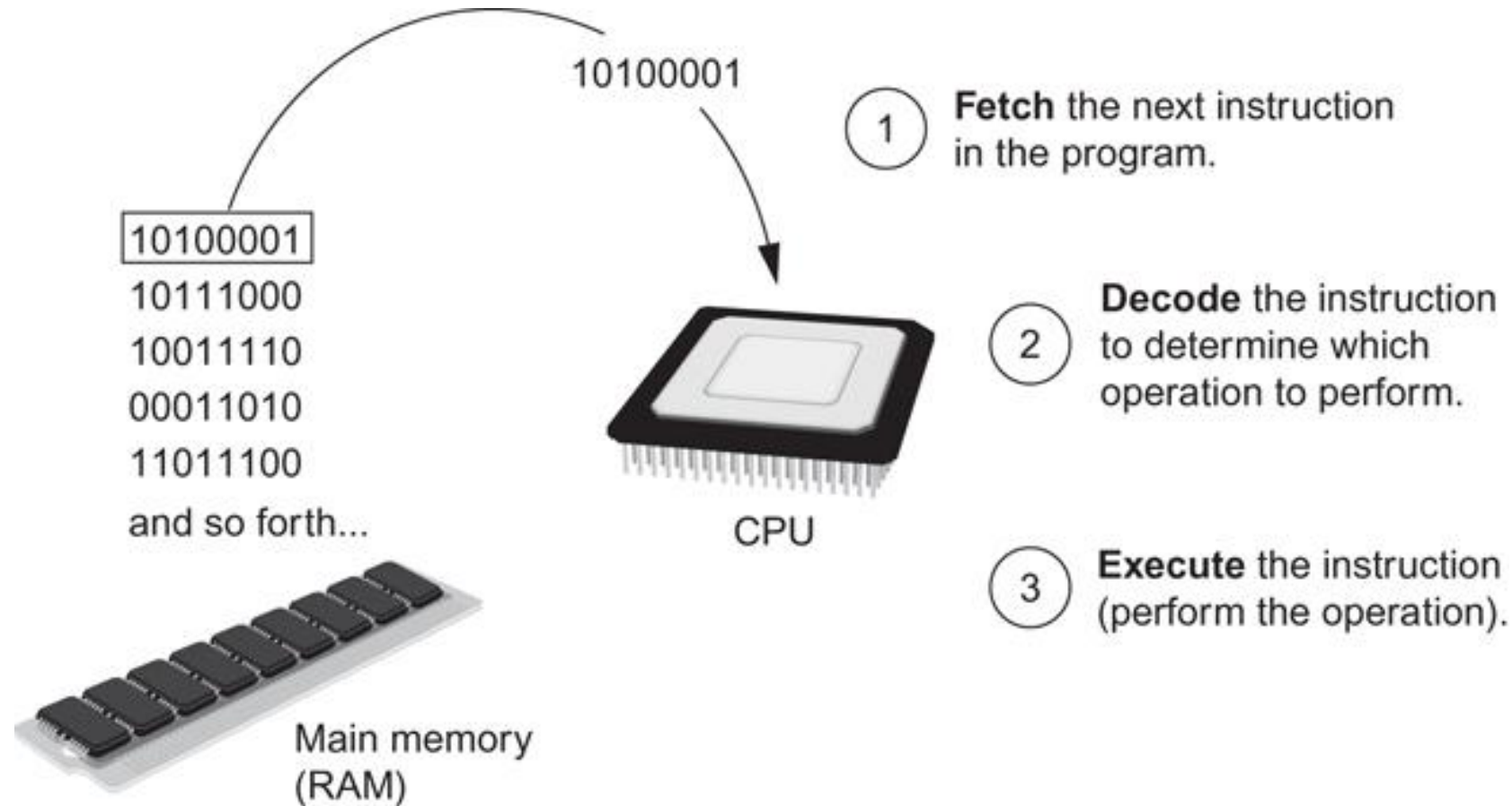


Figure 1-16 The fetch-decode-execute cycle

From Machine Language to Assembly Language

- Impractical for people to write in machine language
- Assembly language: uses short words (mnemonics) for instructions instead of binary numbers
 - Easier for programmers to work with
- Assembler: translates assembly language to machine language for execution by CPU

High-Level Languages

- Low-level language: close in nature to machine language
 - Example: assembly language
- High-Level language: allows simple creation of powerful and complex programs
 - No need to know how CPU works or write large number of instructions
 - More intuitive to understand

Keywords, Operators, and Syntax: an Overview

- Keywords: predefined words used to write program in high-level language
 - Each keyword has specific meaning
- Operators: perform operations on data
 - Example: math operators to perform arithmetic
- Syntax: set of rules to be followed when writing program
- Statement: individual instruction used in high-level language

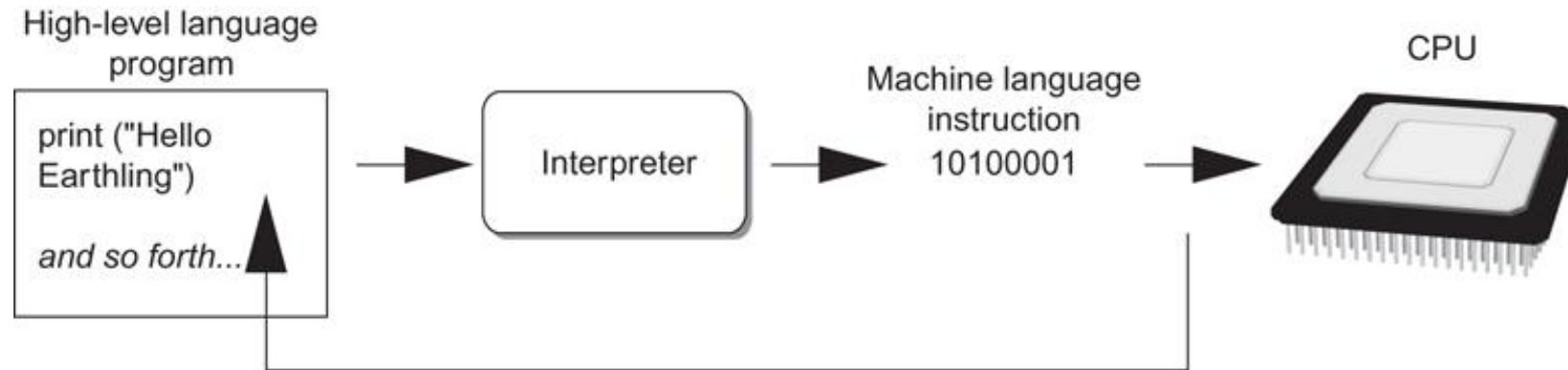
Compilers and Interpreters (1 of 3)

- Programs written in high-level languages must be translated into machine language to be executed
- Compiler: translates high-level language program into separate machine language program
 - Machine language program can be executed at any time

Compilers and Interpreters (2 of 3)

- Interpreter: translates and executes instructions in high-level language program
 - Used by Python language
 - Interprets one instruction at a time
 - No separate machine language program
- Source code: statements written by programmer
 - Syntax error: prevents code from being translated

Compilers and Interpreters (3 of 3)



The interpreter translates each high-level instruction to its equivalent machine language instructions then immediately executes them.

This process is repeated for each high-level instruction.

Figure 1-19 Executing a high-level program with an interpreter

Python

Il Python

È un linguaggio di programmazione:

- Interpretato
- Di alto livello
- Orientato agli oggetti
- Semplice da imparare e usare
- Potente e produttivo
- Ottimo anche come primo linguaggio (molto simile allo pseudocodice)
- Estensibile
- Con una tipizzazione forte e dinamica

Inoltre:

- È open source (www.python.org)
- È multiplatforma
- È facilmente integrabile con C/C++ e Java

Breve storia del Python

Python è stato creato nel 1989 dall'informatico olandese Guido van Rossum (<https://gvanrossum.github.io/>)

Ideato originariamente come linguaggio di scripting, Python si è poi evoluto come linguaggio completo

Il nome fu scelto per via della passione di van Rossum per i Monty Python (<http://www.montypython.com>) e per la loro serie televisiva Monty Python's Flying Circus



Diffusione di Python

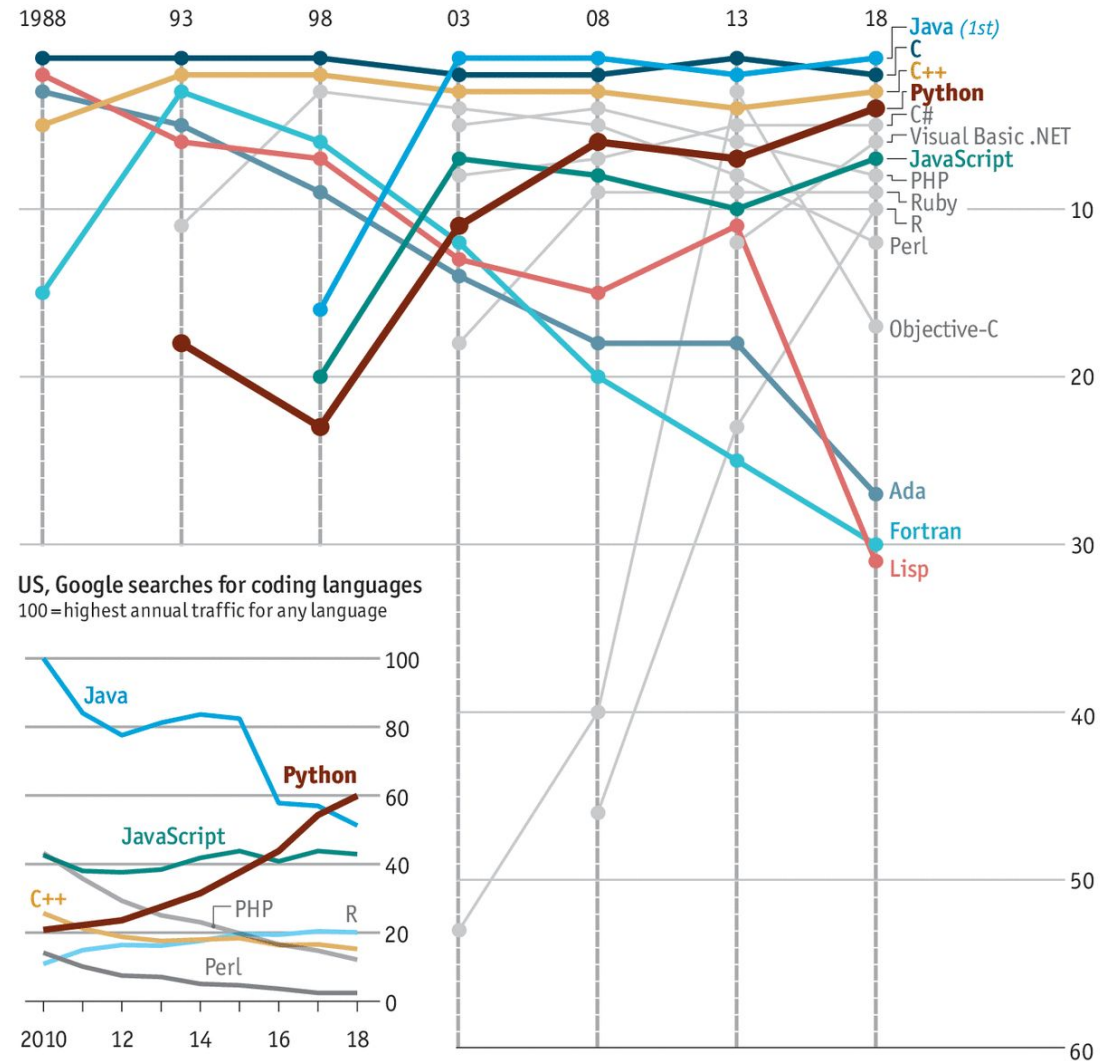
The Economist

Python is becoming the world's most popular coding language

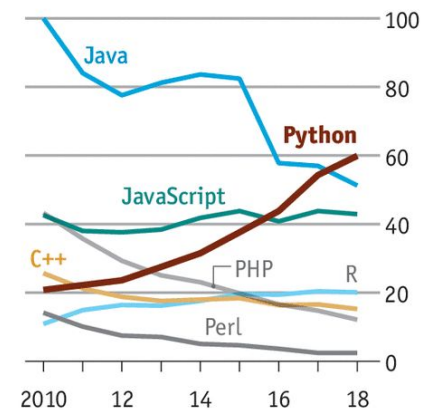
In 2018 Americans have searched for Python on Google more often than for Kim Kardashian, a reality-TV star.

Code of conduct

Ranking of programming languages*



US, Google searches for coding languages
100 = highest annual traffic for any language



Source: TIOBE, Google Trends

*Ranked by global search-engine popularity

The Economist

<https://www.economist.com/graphic-detail/2018/07/26/python-is-becoming-the-worlds-most-popular-coding-language>

Versioni di Python

Sono disponibili due versioni principali di Python:

- Python 2
- Python 3



In questo corso utilizzeremo Python 3

Usare Python

- Python deve essere **installato** per poter essere usato
- Python può essere installato su macchine **Windows**, **Linux** e **Mac** scaricando la versione desiderata dalla pagina ufficiale dei download
<https://wiki.python.org/moin/BeginnersGuide/Download>
- In alternativa, è possibile utilizzare distribuzioni di Python che includono già alcuni pacchetti per applicazioni specifiche (per esempio Anaconda <https://docs.anaconda.com/anaconda/> include pacchetti per applicazioni scientifiche)

Python Interpreter

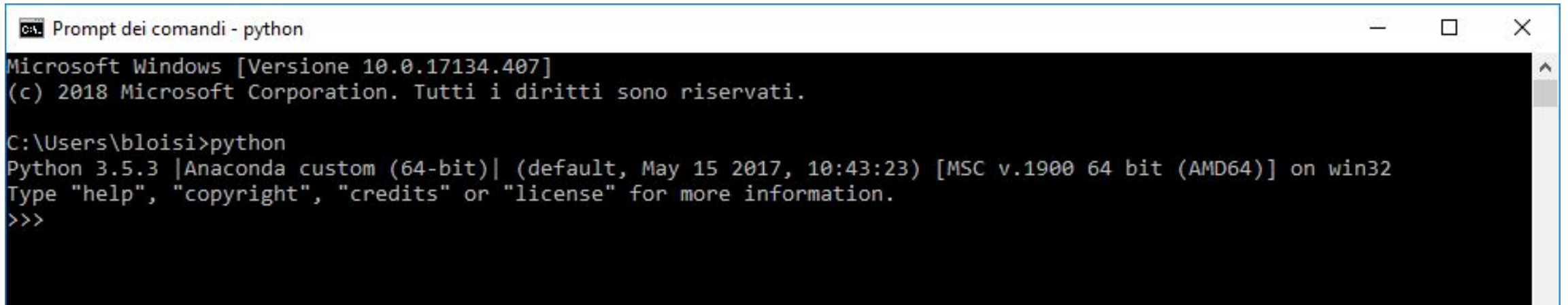
- One of the items installed is the Python interpreter
- Python interpreter can be used in two modes:
 - Interactive mode: enter statements on keyboard
 - Script mode: save statements in Python script

Interactive Mode

- When you start Python in interactive mode, you will see a prompt
 - Indicates the interpreter is waiting for a Python statement to be typed
 - Prompt reappears after previous statement is executed
 - Error message displayed If you incorrectly type a statement
- Good way to learn new parts of Python

Attivare l'interactive mode

Per attivare l'interprete possiamo digitare la parola `python` al prompt di una shell su un PC in cui è installato Python



```
CA Prompt dei comandi - python
Microsoft Windows [Versione 10.0.17134.407]
(c) 2018 Microsoft Corporation. Tutti i diritti sono riservati.

C:\Users\bloisi>python
Python 3.5.3 [Anaconda custom (64-bit)] (default, May 15 2017, 10:43:23) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Apparirà il simbolo `>>>` indicante che l'interprete è pronto a ricevere comandi

Interprete interattivo del Python

L'interprete è un file denominato

- “python” su Unix
- “python.exe” su Windows apre una console
- “pythonw.exe” su Windows non apre una console

Se invocato senza argomenti, presenta una interfaccia interattiva

I comandi si possono inserire direttamente dallo standard input

- Il prompt è caratterizzato da “>>>”
- Se un comando si estende sulla riga successiva è preceduto da “..”

I file sorgente Python sono file di testo, generalmente con estensione “.py”

Modalità interprete

Digitando dei comandi Python in modalità interprete interattivo si ottiene subito una risposta:

```
>>> 5 * 3
```

```
15
```

```
>>>
```

```
>>> a = 5
```

```
>>> b = 6
```

```
>>> 2 * (a+b) + 3*a
```

```
37
```

Writing Python Programs and Running Them in Script Mode

- Statements entered in interactive mode are not saved as a program
- To have a program use script mode
 - Save a set of Python statements in a file
 - The filename should have the `.py` extension
 - To run the file, or script, type
`python filename`
at the operating system command line

Script in Python

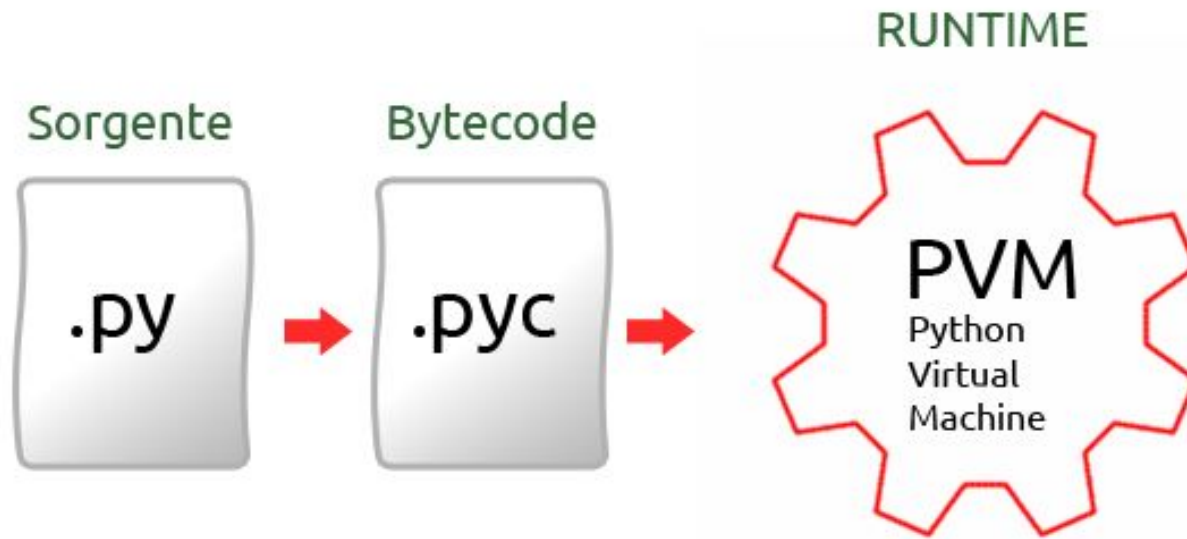
L'interprete interattivo è in grado di leggere e valutare man mano le espressioni inserite dall'utente, ma è anche possibile eseguire script contenenti sequenze di istruzioni Python, digitando il comando:

```
python script.py
```

Ogni volta che viene invocato il comando `python`, il codice scritto viene scansionato per token, ognuno dei quali viene analizzato dentro una struttura logica ad albero che rappresenta il programma.

Tale struttura viene, infine, trasformata in bytecode (file con estensione `.pyc` o `.pyo`). Per potere eseguire questi bytecode, si utilizza un apposito interprete noto come macchina virtuale Python (PVM).

Bytecode Python



Esempio

sorgente (script)

```
def hello()  
    print("Hello, World!")
```

bytecode

2	0	LOAD_GLOBAL	0	(print)
	2	LOAD_CONST	1	('Hello, World!')
	4	CALL_FUNCTION	1	

The IDLE Programming Environment

- IDLE (Integrated Development Program): single program that provides tools to write, execute and test a program
 - Automatically installed when Python language is installed
 - Runs in interactive mode
 - Has built-in text editor with features designed to help write Python programs

IDE per Python



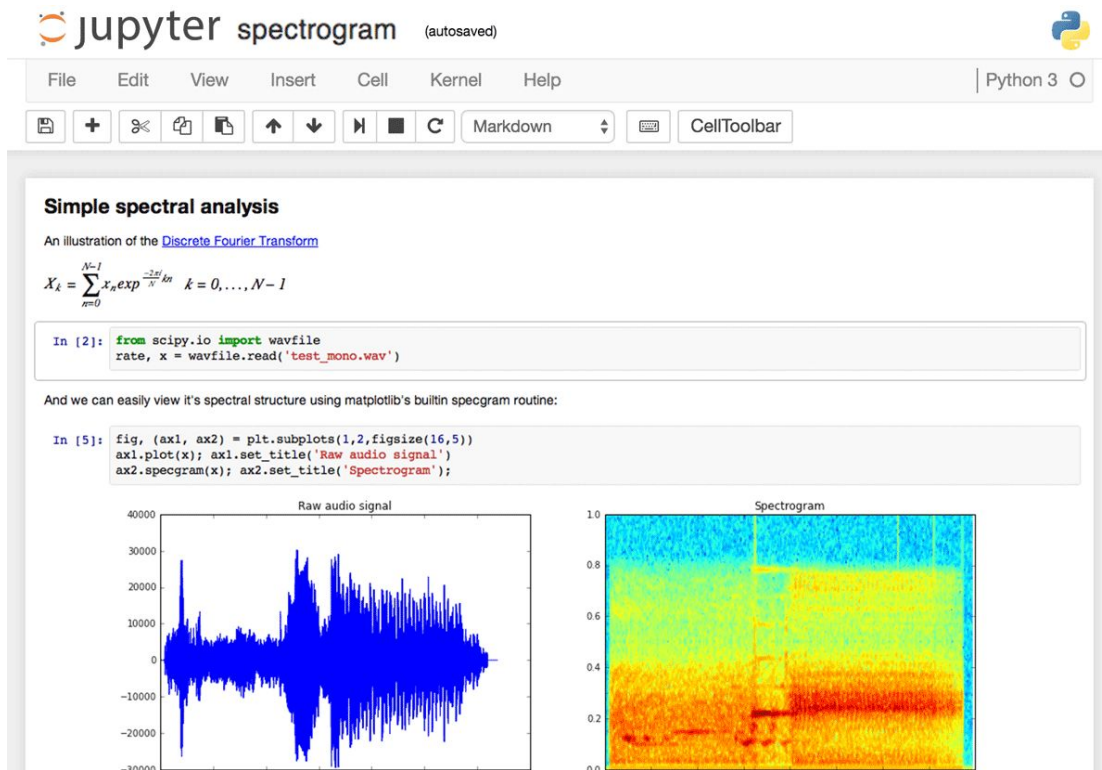
<https://www.jetbrains.com/pycharm/>



<https://www.pydev.org/>

Jupyter Notebook

Jupyter Notebook è un ambiente di sviluppo interattivo, nel quale si possono combinare codice, testo, figure, formule matematiche e video.



Per installare Jupyter Notebook fare riferimento alla pagina <https://jupyter.org/install>



<https://www.dataquest.io/blog/jupyter-notebook-tips-tricks-shortcuts/>

Colab

Google Colab

Colaboratory è un tool per la ricerca specifico per applicazioni di machine learning. È un ambiente Jupyter notebook (<https://jupyter.org/>) che non richiede l'installazione di alcuna libreria poiché **è già pronto per l'uso**.

Colaboratory può essere usato con i più diffusi browser ed è testato sulle versioni desktop di **Chrome** e **Firefox**.

Colaboratory è un progetto di ricerca che viene distribuito in accordo alla licenza BSD 3-Clause "New" or "Revised" License.

Tutti i file Colaboratory sono memorizzati in **Google Drive**.



Google Colab

Colaboratory supporta Python 3 e ha smesso di supportare Python 2 dal 1° gennaio 2020

Il codice viene eseguito su una macchina virtuale dedicata collegata all'account Google.

Google impone dei limiti temporali per l'uso continuativo della macchina virtuale (12 ore)



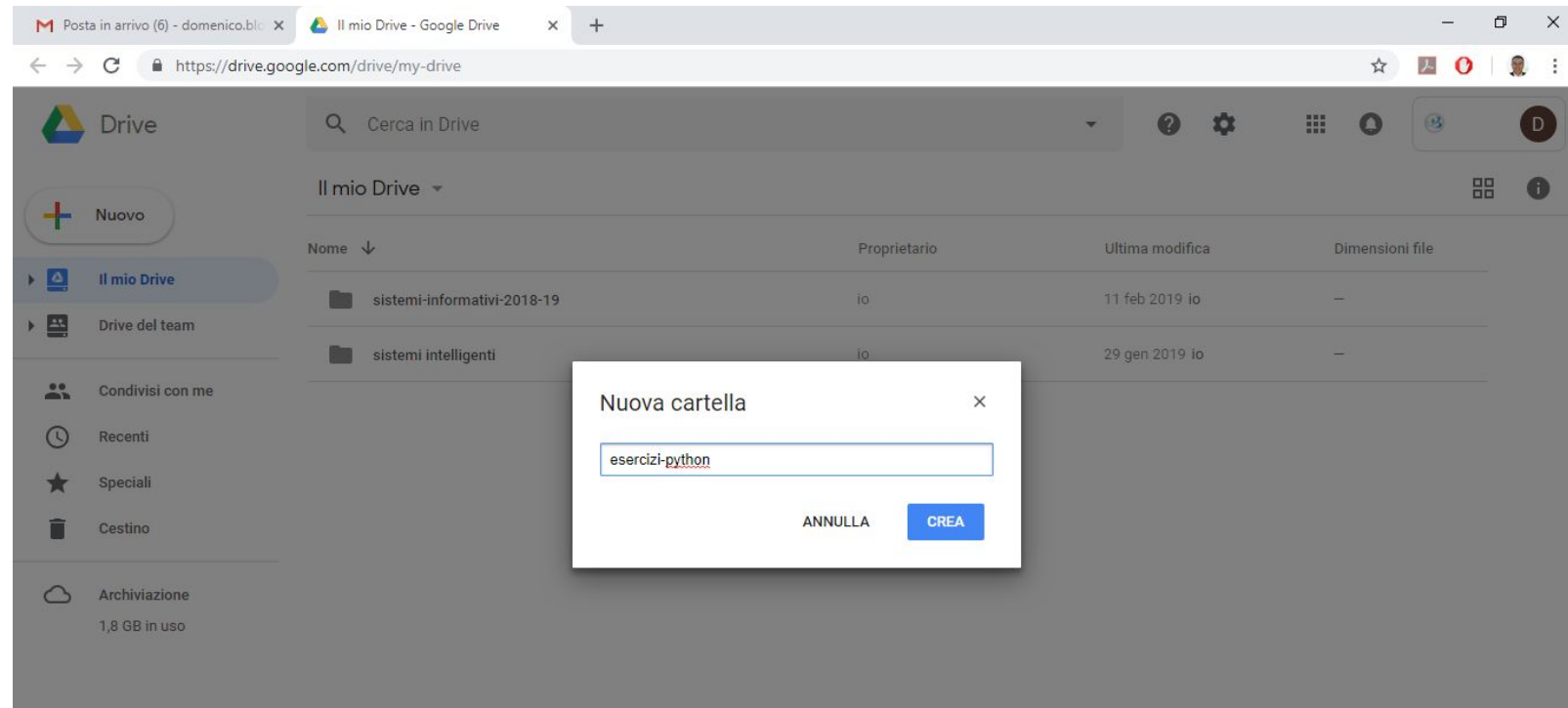
<https://research.google.com/colaboratory/faq.html>

Primo programma Python con Google Colab

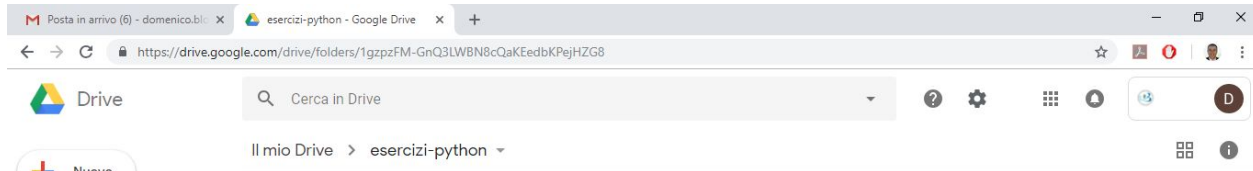


1. Apriamo il nostro Google Drive

2. Creiamo una nuova cartella denominata per esempio 'esercizi-python'



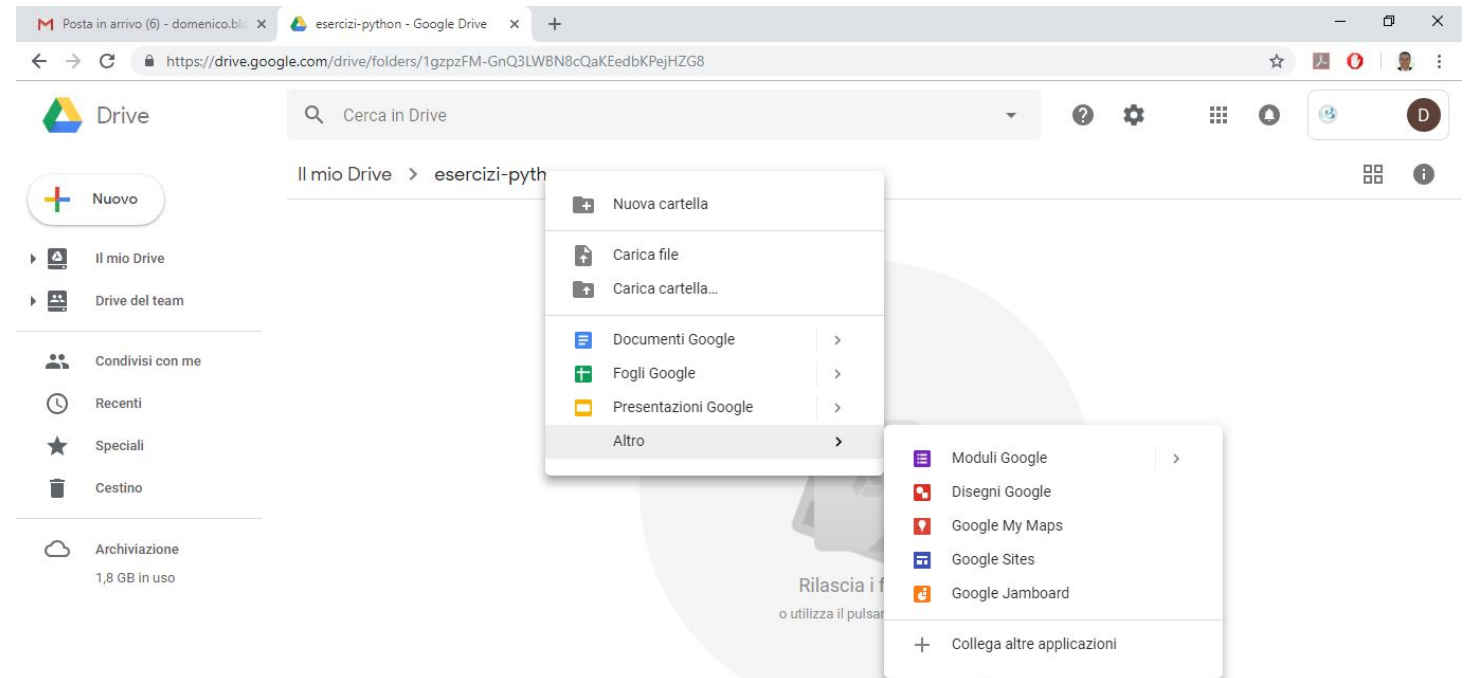
Creazione del primo file sorgente



3. Spostiamoci
nella cartella
'esercizi-python'

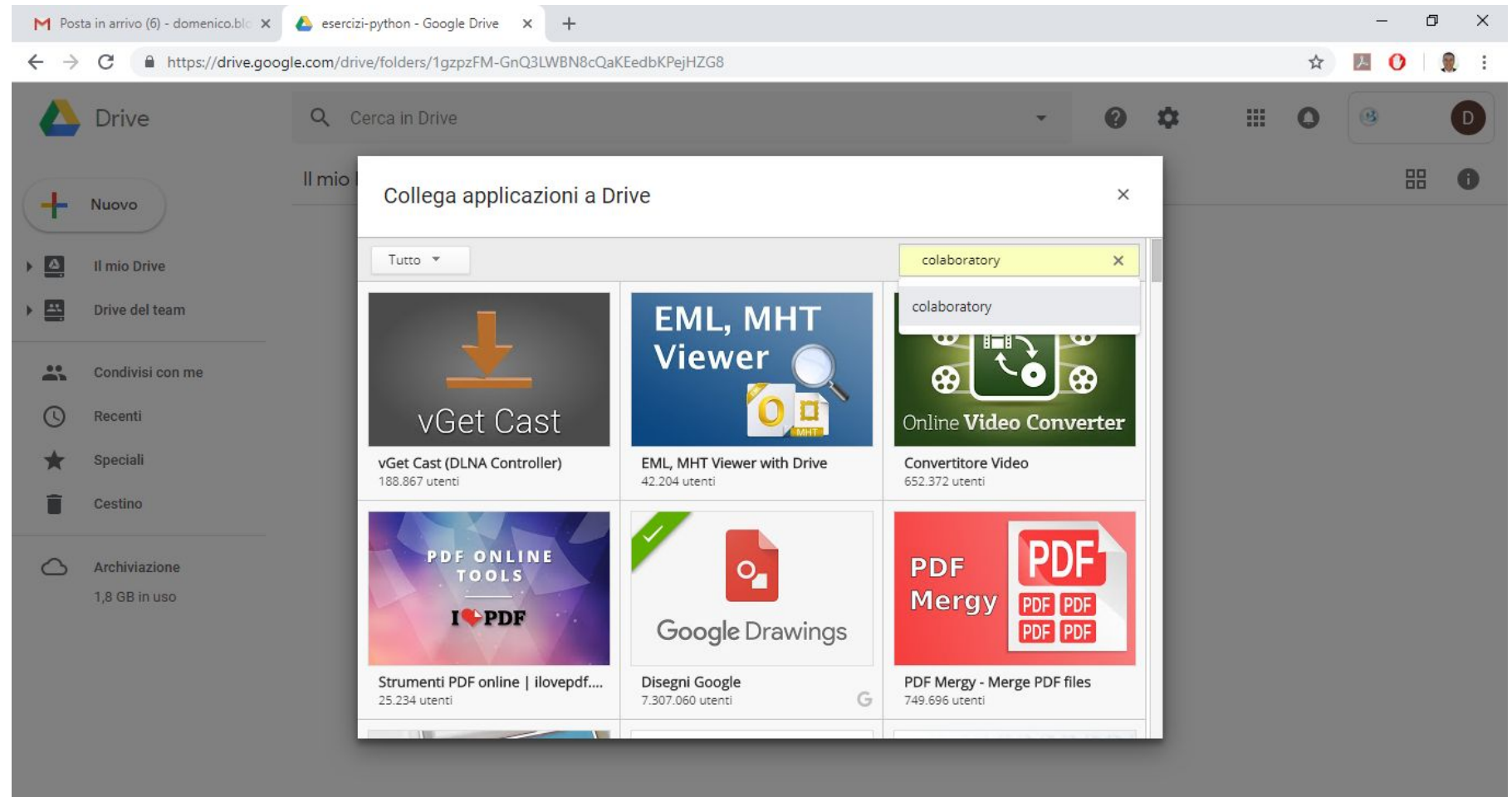


4. Usando il tasto destro del
mouse, selezioniamo 'Altro' e
poi 'Collega altre applicazioni'



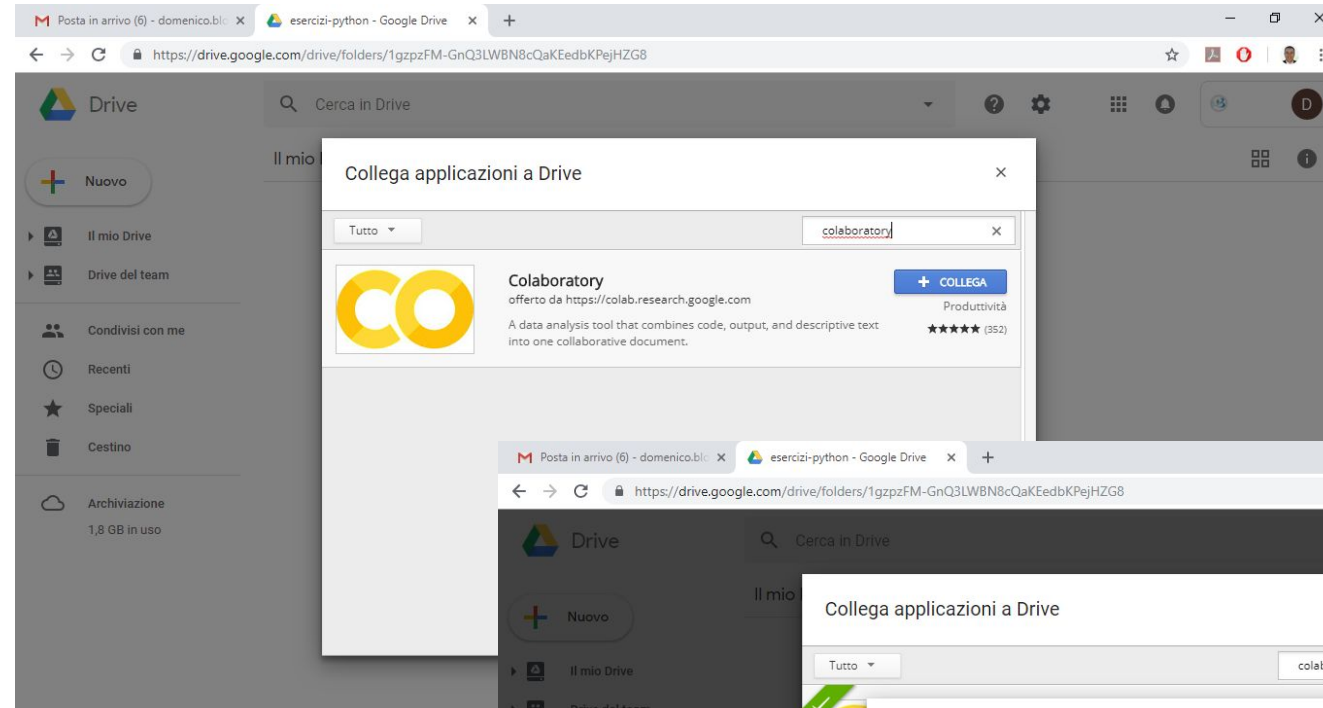
Ricerca applicazione Google Colaboratory

5. Nella casella a destra inserire la parola 'colaboratory' per poter collegare l'applicazione Google Colaboratory al proprio Google Drive. Premere invio

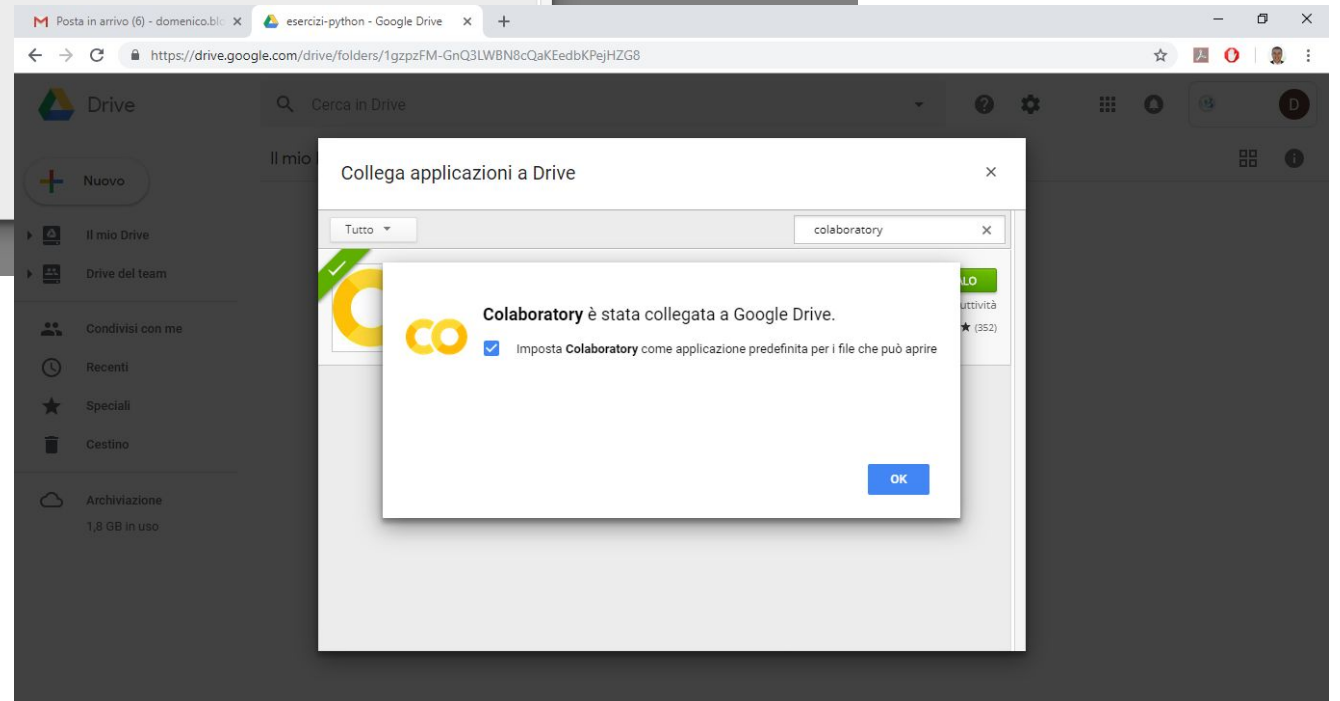


Colleghiamo l'applicazione al nostro Google Drive

6. Selezionare l'opzione 'collega'

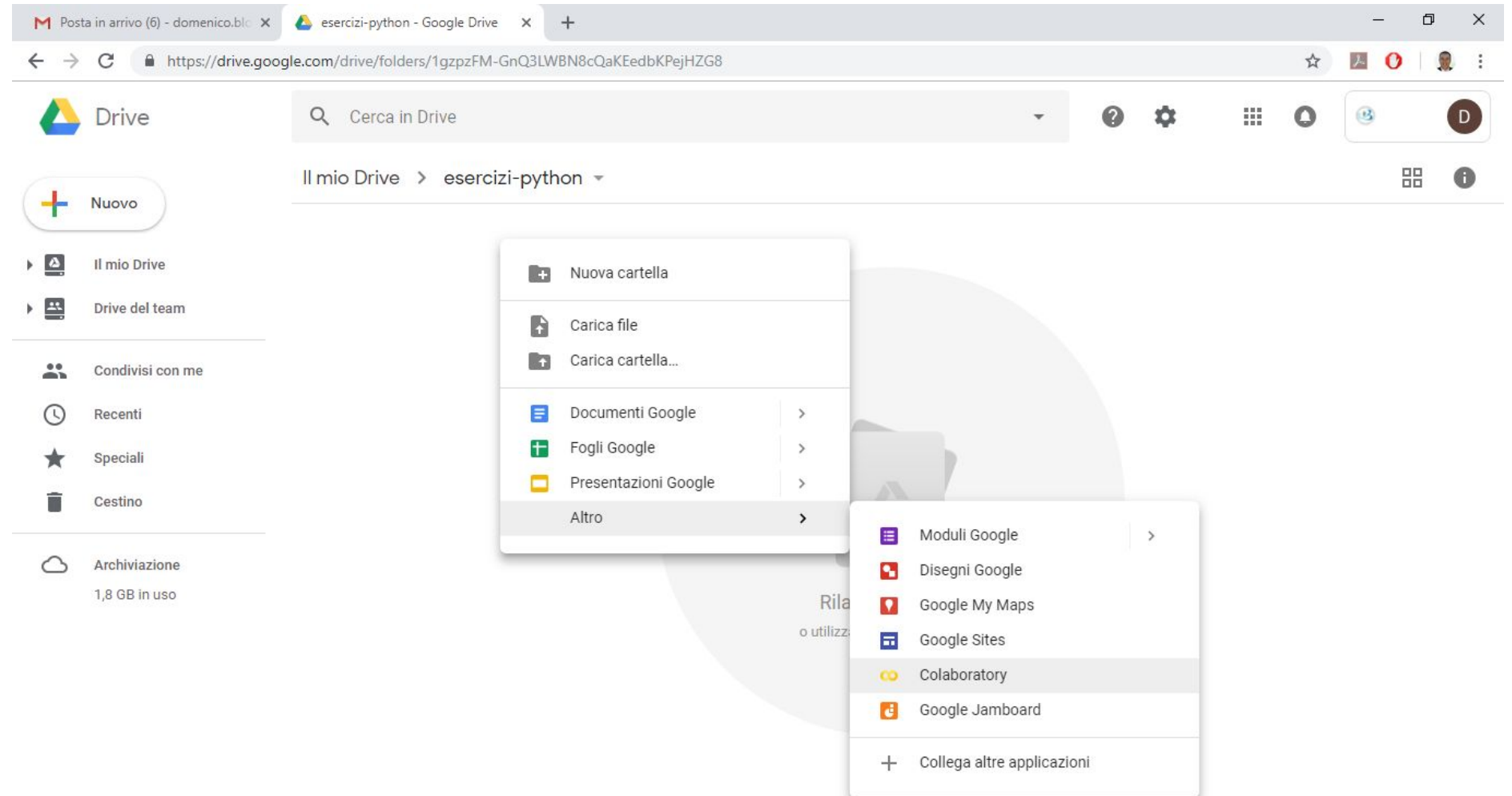


7. Selezionare 'OK'



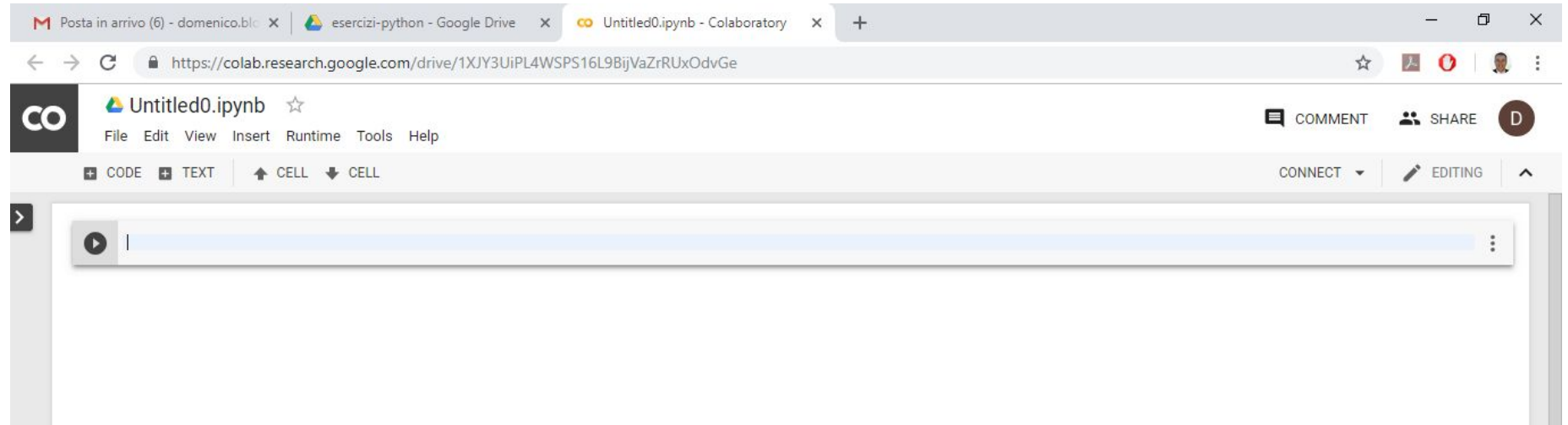
Selezioniamo l'applicazione Colaboratory

8. Usiamo il tasto destro del mouse per selezionare 'Altro' e poi troveremo la nuova voce 'Colaboratory'

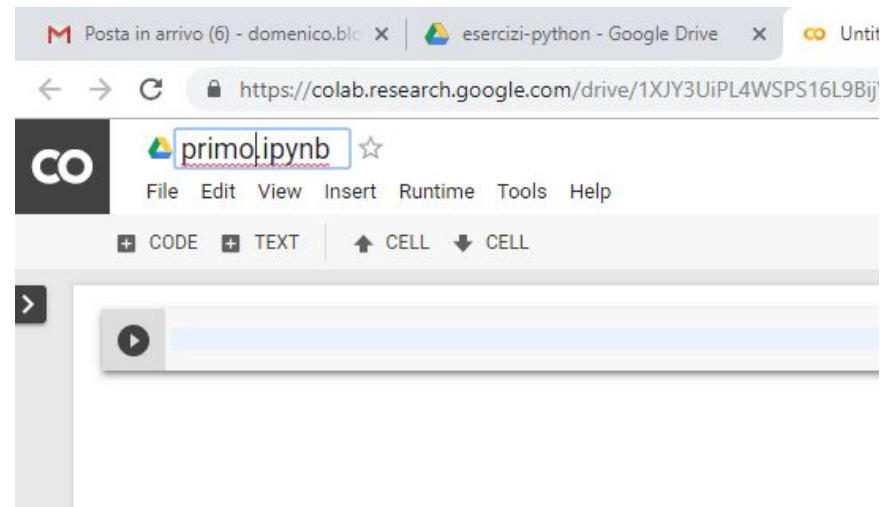


Creiamo un file Colaboratory

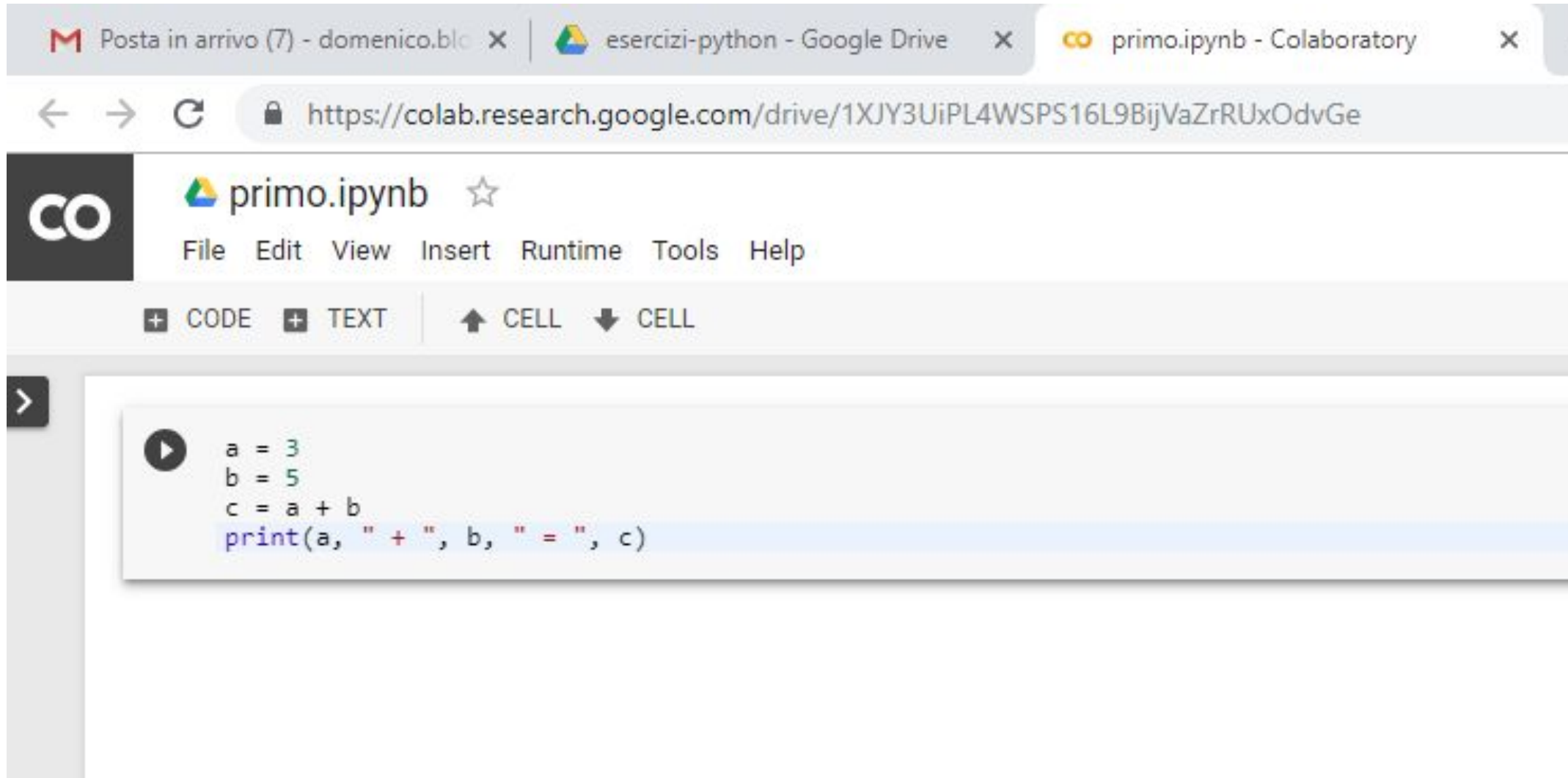
9. Verrà creato un file 'Untitled0.ipynb' che possiamo usare per creare il nostro primo programma in Python



10. Rinominiamo il file come 'primo.ipynb'



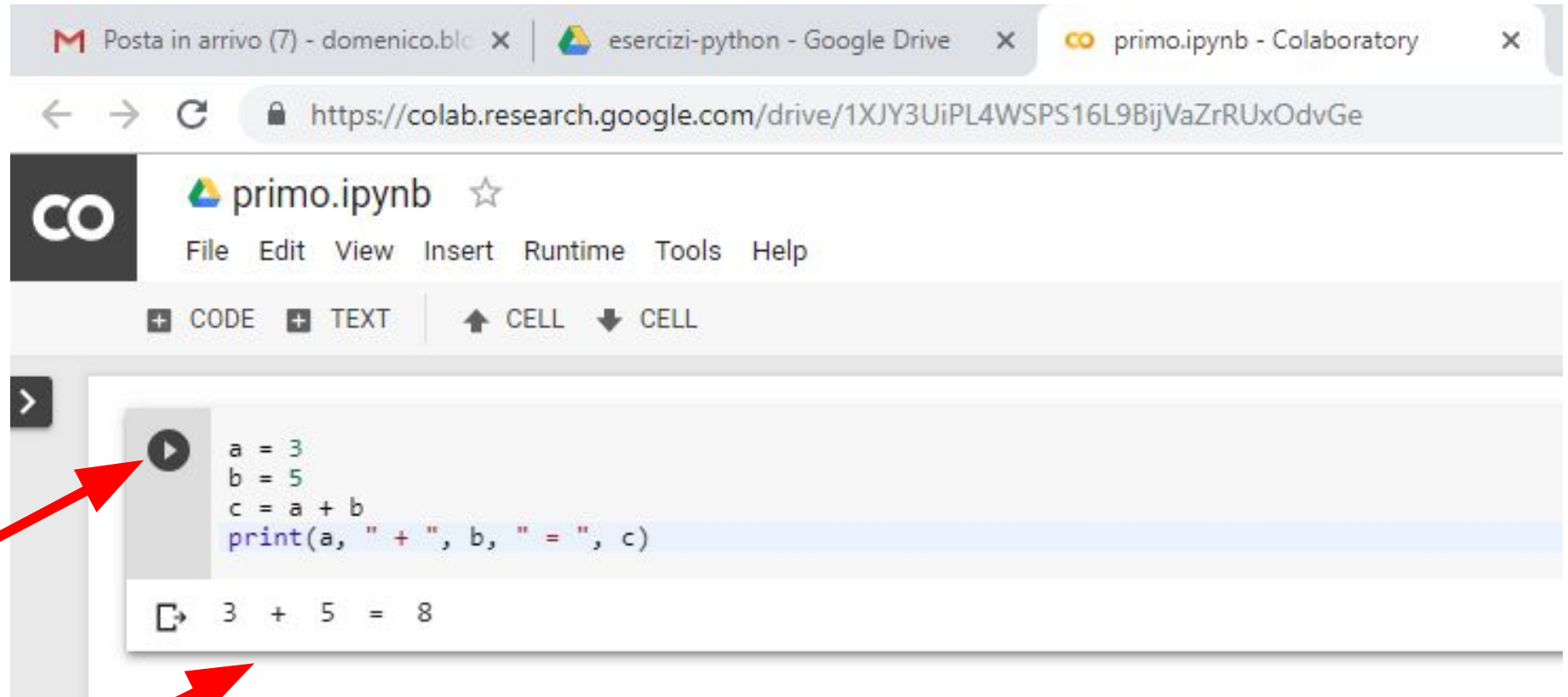
Scriviamo il primo programma Colab



The screenshot shows a web browser with three tabs: "Posta in arrivo (7) - domenico.bl...", "esercizi-python - Google Drive", and "primo.ipynb - Colaboratory". The address bar shows the URL "https://colab.research.google.com/drive/1XJY3UiPL4WSPS16L9BijVaZrRUxOdvGe". The notebook interface includes a menu bar with "File", "Edit", "View", "Insert", "Runtime", "Tools", and "Help". Below the menu bar are buttons for "+ CODE", "+ TEXT", "↑ CELL", and "↓ CELL". The main area contains a code cell with a play button icon and the following Python code:

```
a = 3
b = 5
c = a + b
print(a, " + ", b, " = ", c)
```

Esecuzione del primo programma Colab



The screenshot shows a web browser with three tabs: 'Posta in arrivo (7) - domenico.blo', 'esercizi-python - Google Drive', and 'primo.ipynb - Colaboratory'. The address bar shows the URL 'https://colab.research.google.com/drive/1XJY3UiPL4WSPS16L9BijVaZrRUxOdvGe'. The notebook interface includes a menu bar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', and 'Help'. Below the menu bar are buttons for '+ CODE', '+ TEXT', and 'CELL' (with up and down arrows). A code cell is selected, containing the following Python code:

```
a = 3
b = 5
c = a + b
print(a, " + ", b, " = ", c)
```

The output of the cell is displayed below the code: '3 + 5 = 8'. Two red arrows point to the 'Run' button (a play icon) and the output text.

Selezionando 'Run cell' possiamo eseguire il nostro codice e ottenere l'output

Corso di *STATISTICA, INFORMATICA, ELABORAZIONE DELLE INFORMAZIONI*

Modulo di Sistemi di Elaborazione delle Informazioni

Introduzione alla programmazione



UNIVERSITÀ DEGLI STUDI DELLA BASILICATA



Docente:
Domenico Daniele Bloisi

