Programmi

Informatica@SEFA 2018/2019 - Lezione 3

Massimo Lauria <massimo.lauria@uniroma1.it>
http://massimolauria.net/courses/infosefa2018/

Venerdì, 28 Settembre 2018

Questionario

Questo è lo stesso questionario che ho indicato alla fine della lezione scorsa.

bit.ly/INFO2018-02d

Decodificate il testo codificato in ASCII

ASCII Code Chart

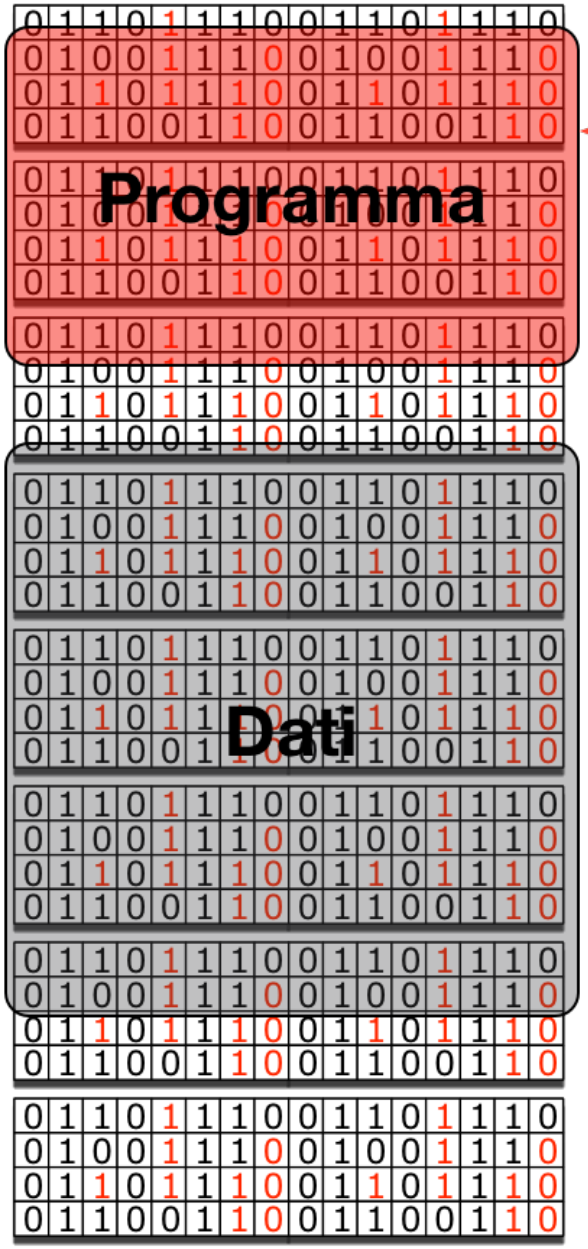
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2		!		#	$	%	&		(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

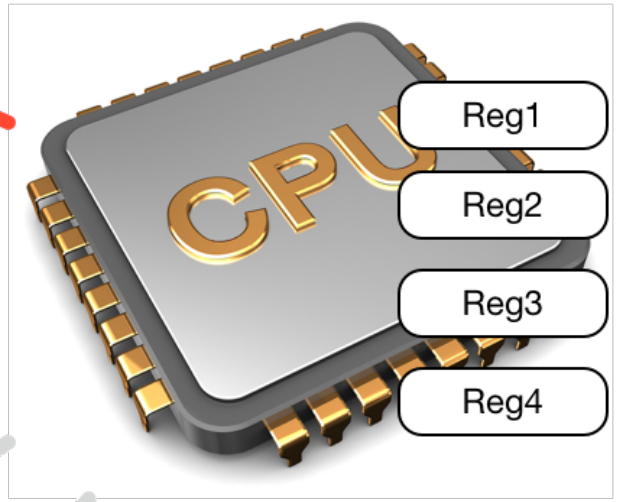
Figure: Tabella ASCII (fonte:Wikipedia)

[76, 101, 103, 103, 101, 116, 101, 32, 105, 108, 32, 109, 97, 116, 101, 114, 105, 97, 108, 101, 32, 80, 82, 73, 77, 65, 32, 100, 105, 32, 118, 101, 110, 105, 114, 101, 32, 97, 32, 108, 101, 122, 105, 111, 110, 101, 33]

Programmazione dei computer

Linguaggio macchina per CPU





1. RAM <-> Registri
   4 o 8 byte
2. Registri <-> Registri

3. Operazioni matematiche
   o logiche tra registri

4. saltare nel programma

Osservazione

Il programma è finito, ma può lavorare su quantità di dati potenzialmente infinita. Questo è possibile grazie alle **istruzioni di salto** della CPU. Ad esempio

Salto assoluto

- E.g., salta alla pos. 531 del programma

Salto condizionato

- E.g., salta alla pos. 421 se il terzo registro è 0

Linguaggi di programmazione evoluti

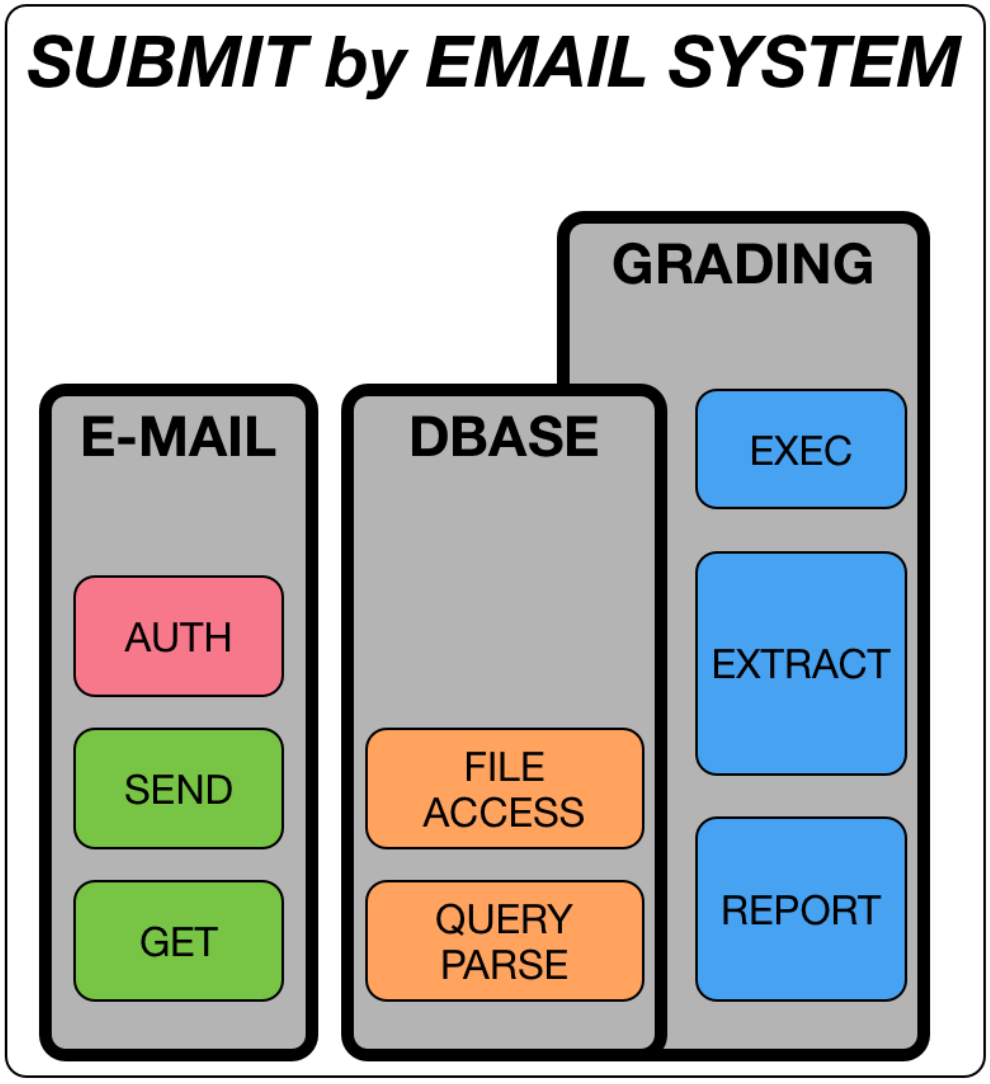
Vogliamo programmare così

```
for x in [1,2,3,4,5]:                    1
    print(x)                             2
```

invece di programmare in linguaggio macchina

```
entrypoint:                              1
        movq    %rdi, -8(%rbp)           2
        movq    -8(%rbp), %rdi           3
        cmpq    $0, 80(%rdi)             4
        sete    %al                      5
label2:                                  6
        xorb    $-1, %al                 7
        andb    $1, %al                  8
        movzbl  %al, %ecx                9
        movslq  %ecx, %rdi               10
```

Astrazione e sotto-problemi



- gerarchia organizzativa
- sotto-problemi e sotto-programmi
- nascondere dettagli
- interfacce
- facile da analizzare
- divisione del lavoro

Astrazioni e organizzazione del pensiero

Le astrazioni sono dei **pezzi logici** che modellano elementi del problema analizzato.
Sono gradini per costruire astrazioni di livello più alto.

Strumenti per le astrazioni

- **Sistema operativo**: dispositivi di I/O, multiprocessi
- **Librerie** (libraries): sotto programmi altrui
- **Elementi del linguaggio**: costruire le proprie astrazioni

Linguaggi di alto e basso livello

Script > L. Applicazioni > L. di Sistema > L. Macchina

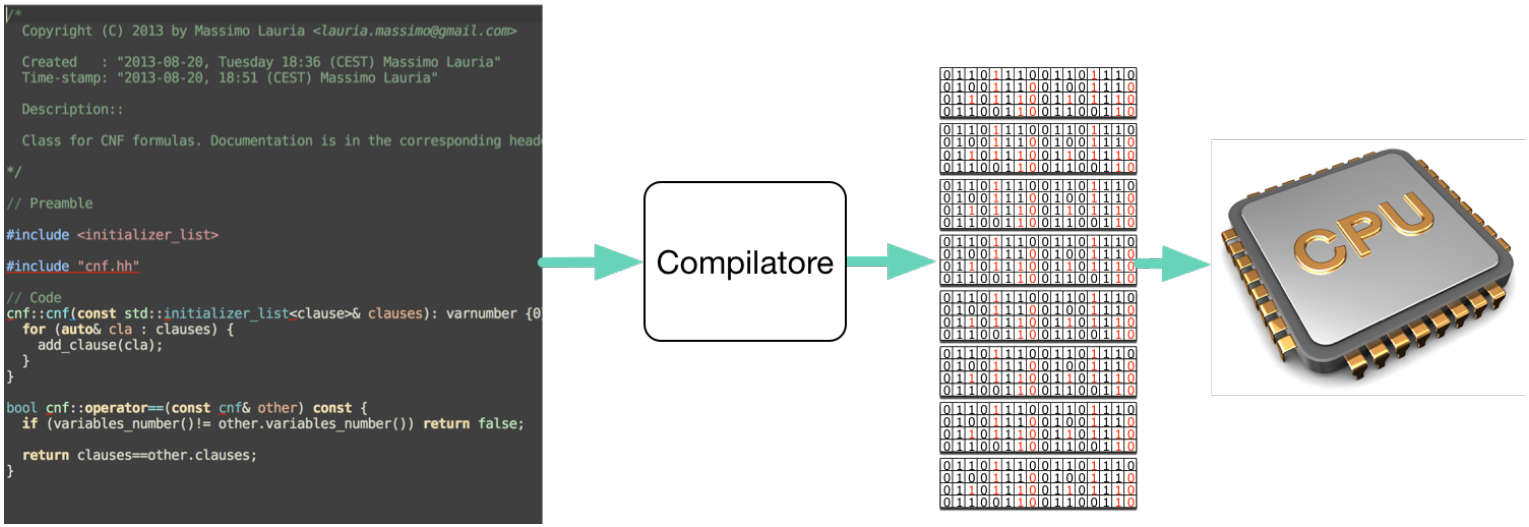
Alto livello

- astrazioni più potenti/espressive
- più facili
- meno efficienti

Basso livello

- astrazioni meno potenti
- più difficili
- più efficienti

Traduzione in blocco (e.g. C, C++)

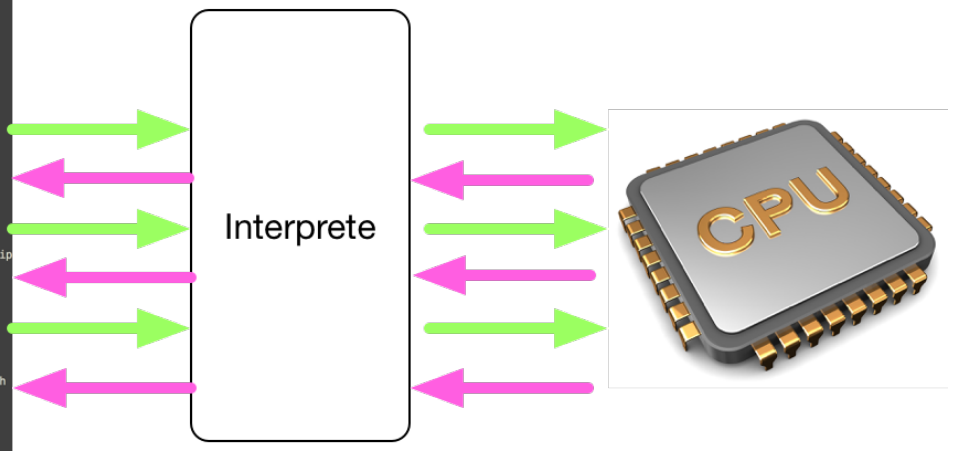


Il programma viene tradotto/ottimizzato in linguaggio macchina, da un **compilatore**, pronto per essere eseguito dalla CPU

- più sicuri
- più efficienti
- meno flessibili
- ling. di alto e basso livello

Esecuzione interattiva (e.g. Python)

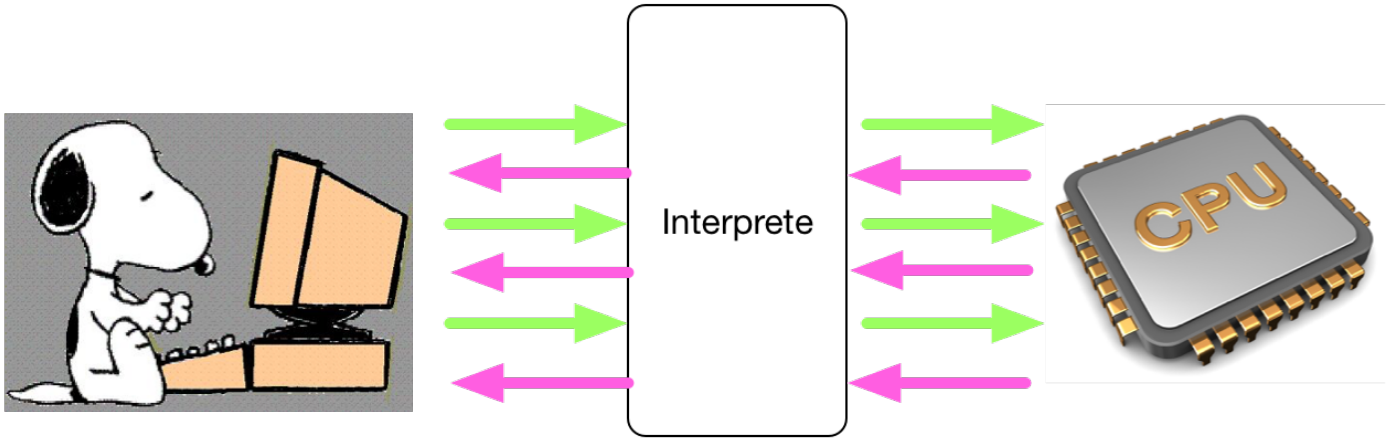
```
r"""Components for command line interface

CNFgen has many command line entry points to its functionality, and
some of them expose the same functionality over and over. This module
contains useful common components.

Copyright (C) 2012, 2013, 2014, 2015, 2016  Massimo Lauria <lauria@kth.se>
https://github.com/MassimoLauria/cnfgen.git

"""

from __future__ import print_function


import sys
import argparse

import networkx
import random

from itertools import combinations, product

from .graphs import supported_formats as graph_formats
from .graphs import readGraph,writeGraph
from .graphs import bipartite_random_left_regular,bipartite_random_regular,bip
ft
from .graphs import bipartite_sets
from .graphs import dag_complete_binary_tree,dag_pyramid
from .graphs import sample_missing_edges


try: # NetworkX >= 1.10

    complete_bipartite_graph    = networkx.bipartite.complete_bipartite_graph
    bipartite_random_graph      = networkx.bipartite.random_graph
    bipartite_gnmk_random_graph = networkx.bipartite.gnmk_random_graph

except AttributeError: # Networkx < 1.10

    from networkx import complete_bipartite_graph
    from networkx import bipartite_random_graph
    from networkx import bipartite_gnmk_random_graph


__all__ = [ "register_cnfgen_subcommand","is_cnfgen_subcommand",
            "DirectedAcyclicGraphHelper", "SimpleGraphHelper", "BipartiteGraph
```



Il programma viene letto da un **interprete** che esegue passo passo quello che è scritto nel programma.

- meno sicuri
- meno efficienti
- più flessibili
- ling. di alto livello

Esecuzione interattiva (e.g. Python)



Il programma viene letto da un **interprete** che esegue passo passo quello che è scritto nel programma.

- meno sicuri
- meno efficienti
- più flessibili
- ling. di alto livello

Python – presentazione ufficiale

```
A = [1,2,3,4,5,6,7,8,9,10]                                    1
B = [ x*x for x in A ]                                        2
print(B)                                                      3
```

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

Python è un linguaggio ad alto livello

- semplice
- libreria molto ricca di funzioni
- interattivo
- più lento di molti altri linguaggi

Python di alto livello (e.g., i numeri)

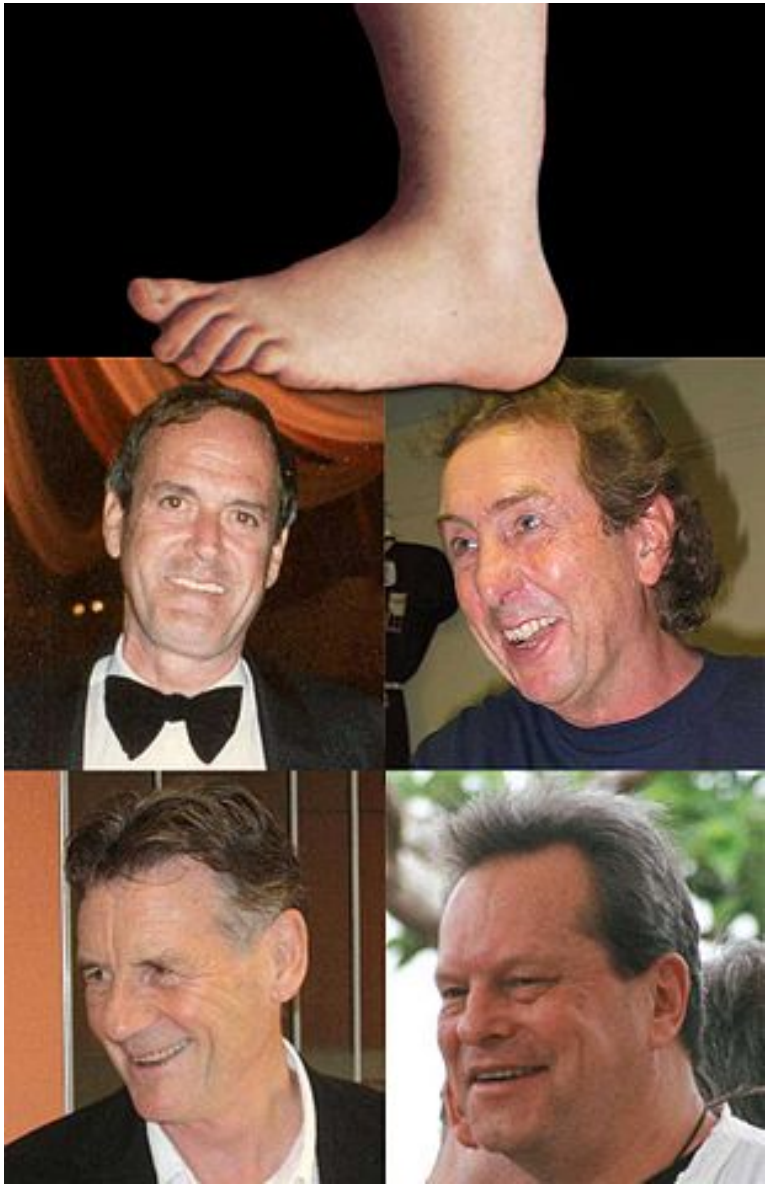
Per esempio Python ha numeri di dimensione arbitraria

- nasconde i dettagli della CPU
- gestisce gli *overflow*

Più ad alto livello di C,C++

- stessi numeri della CPU
- incompatibilità su CPU diverse

Python - risorse (in inglese)



Link utili:

- `http://www.pythontutor.com/`
- `https://docs.python.org/3/`

Strumenti:

- IPython `https://ipython.org/`
- Anaconda: `https://www.anaconda.com/`
- Thonny (offline): `http://thonny.org/`
- repl.it (online): `https://repl.it/`

SQL e basi di dati

```
select ID,name,surname from students where enroll='2017'      1
```

ID	name	surname
10231	Mario	Rossi
01234	Giancarlo	Garibaldi
02135	Grace	Hopper
02107	Guybrush	Threepwood
12042	Robert	Wyatt

- richieste dati
- dichiarativo
- standard

Scrittura di codice

Astrazioni

Astrazione*: l'atto di non tenere in considerazione una o più proprietà di un oggetto complesso, così da poter analizzarne altre.*

Variabili

Associano un valore ad un nome. Il valore associato ad un nome può cambiare nel tempo.

```
nome_variabile = expressione                                    1
```

Ad esempio

```
altezza=10                                                      1
larghezza=10                                                    2
                                                                3
area_rettangolo = altezza * larghezza                           4
perimetro_rettangolo = 2*( altezza + larghezza )                5
```

Funzioni

Funzioni: riuso di una **sequenze di istruzioni**

```
def nome_funzione(parametro1, parametro2, ..., parametroN):   1
  istruzione_1                                                2
  istruzione_2                                                3
  ...                                                         4
  istruzione_M                                                5
  return espressione # opzionale                              6
```

Per usare una funzione già creata

```
nome_funzione(valore1, valore2, ..., valoreN)                 1
```

Funzioni: esempio

(Fonte: Cap. 4 del testo di F.Pellacini)

```
def area_cilindro(raggio, altezza):                                    1
    pigreco = 3.14159                                                  2
    area = pigreco * raggio ** 2                                       3
    circonferenza = 2 * pigreco * raggio                               4
    return 2 * area + altezza * circonferenza                          5
                                                                       6
print(area_cilindro(10, 5))                                            7
print(area_cilindro(20, 10))                                           8
```

```
942.477
3769.908
```

- `raggio`, `altezza` sono i **parametri formali**
- `(10,5)` e `(20,10)` sono i **parametri effettivi**

Nomi legali per variabili e funzioni

- può contenere lettere maiuscole o minuscole
- può contenere il carattere _
- può contenere numeri

Non può iniziare con un numero

```
variable1 = 0                                    1
variabile_con_nome_lungo = 0                     2
NomeMaiuscolo = 0                                3
def _my_print():                                 4
  pass                                           5
```

Questionario

bit.ly/INFO2018-03a

Situazione in Aula XV Lab Tiburtina

- Prenotato Aula XVI negli stessi orari
- Dividetevi più o meno a metà
- Farò un tutorial in entrambe le aule
- Mentre mi aspettate potete cominciare a fare l'esercizio seguente.
- Non ho il dono dell'ubiquità: chi ha capito bene le procedure della prima lezione può aiutare gli altri studenti.

Esercizio

Cercate di

1. scrivere il programma di esempio nella slide ”Funzioni: esempio”
2. eseguirlo con python3

Bonus: fate piccole variazioni nel programma e vedete che succede.

Lo scopo è che ognuno debba imparare a scrivere e ad eseguire codice python **in maniera autonoma**.