

# Introduzione a Python



*by Martina Bellio - 2020*

# Introduzione a Python



## Comunicazioni di servizio:

1. Modalità WEBINAR (solo chat attiva),  
ore 10.00-12.00 e 14.00-16.00  
intervalli: 10:50-11:10 ; 14:50:15:10, per la chat
2. COME INTERAGIRE : negli intervalli e alla fine dell'incontro
3. E dopo questa giornata ? E-mail : [canella@lnl.infn.it](mailto:canella@lnl.infn.it)
4. Come è nato questo corso ? ( Raspberry PI, laboratori e stage estivi, corso LNL, corso INFN)
5. A chi devo riconoscimenti ? (ACKNOWLEDGMENTS)  
Ai miei colleghi LNL, alla RASPBERRY PI Foundation,  
a : Dr. Charles R. Severance (MSU)



Il corso ...

Questi incontri hanno lo scopo di fornire una prima introduzione all'uso del linguaggio di programmazione PYTHON.

Durata corso: 2 lezioni di 50' ciascuna, con la descrizione di esempi pratici di brevi programmi.



9 giugno 2020 : Per cominciare ...

(mattina) Nuts and Bolts

9 giugno 2020 : Programmi object-oriented

(pomeriggio) Esempi



## 1 - Per cominciare ...

- Introduzione a Python.
- Installazione ed uso dell'ambiente IDLE.
- Primi esempi (“Ciao Mondo!”).
- Uso interattivo e tramite script.
- Uso diretto dell'interprete Python.



## Nuts and Bolts

- I costrutti base per i programmi procedurali.
- Le variabili e i valori.
- Input/Output.
- Le strutture di controllo.
- Esempi ed esercizi.



## 2 - I programmi object-oriented (OO)

- Introduzione alla programmazione OO.
- Predisposizione di IDLE per usare GUIZERO.
- Elementi base per programmi OO con Guizero.
- Primi esempi di programmi Guizero.
- Idee per proseguire con Guizero.



## Esempi (di programmi object-oriented - OO)

- GUI : una finestra con un testo.
- Una finestra con un pulsante che modifica la finestra stessa.
- Un text editor.
- Esempi per proseguire.





## Per Finire ...

- Cos'altro si può fare con Python ?
- Gestione di GUI e grafica in generale.
- Tools di comunicazione in rete.
- Tools di supporto alla gestione di siti web.
- Elaborazioni di immagini.



## Suggerimenti ...

Dove seguire corsi dedicati:

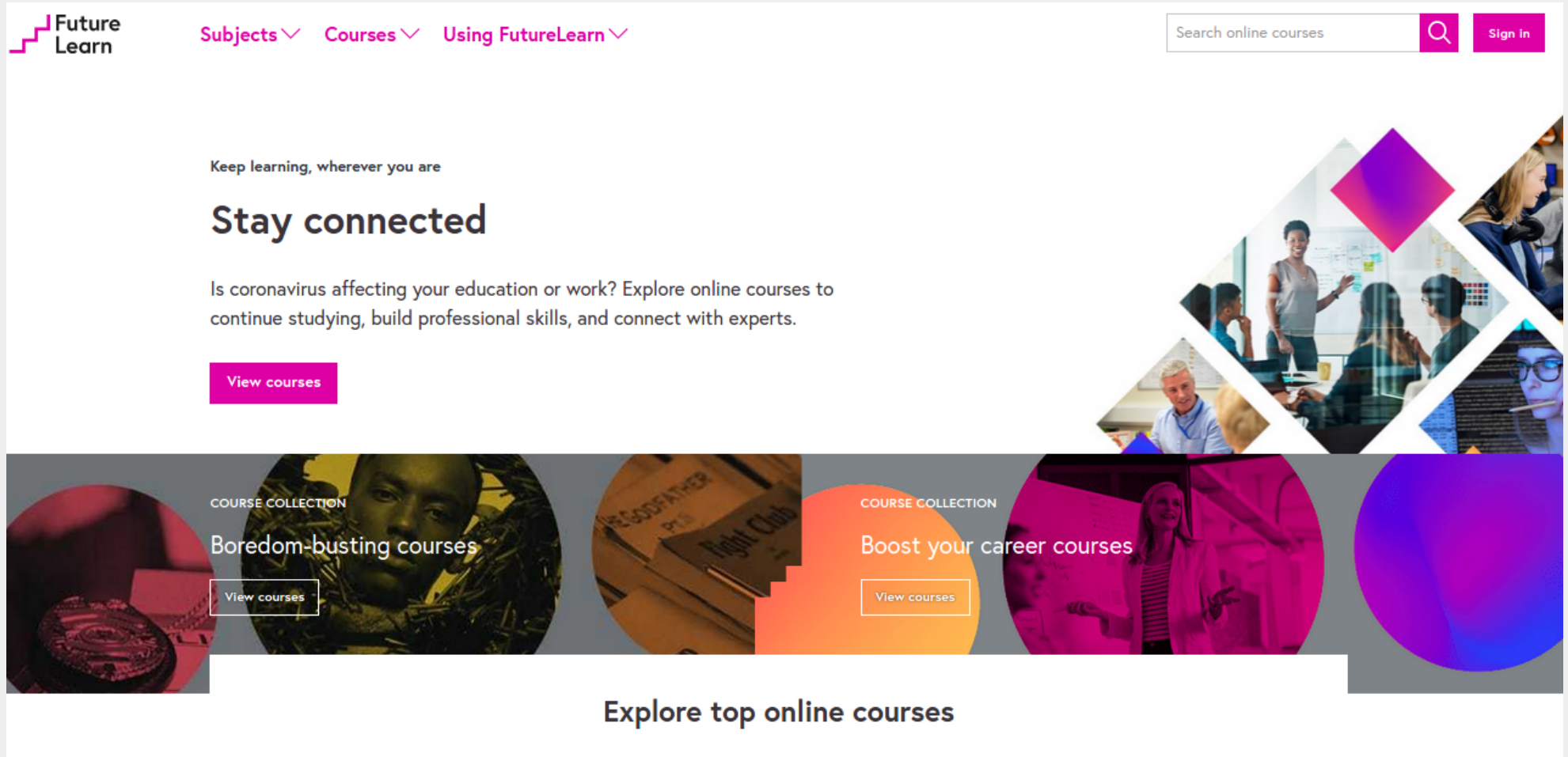
Su <https://www.futurelearn.com> (corsi brevi)

- Raspberry Pi Foundation : 8 corsi (101, 102, 103)  
3 o 4 settimane;

- **Programming for Everybody** (Univ. Michignan)  
6 settimane.

Exploring Data Using Python 3, by Dr. Charles R. Severance (MSU)

# Altre risorse 1



The screenshot shows the FutureLearn website homepage. At the top left is the FutureLearn logo. Navigation links for 'Subjects', 'Courses', and 'Using FutureLearn' are visible. A search bar on the right contains the text 'Search online courses' and a 'Sign in' button. The main content area features the headline 'Stay connected' with the subtext 'Keep learning, wherever you are'. Below this is a paragraph: 'Is coronavirus affecting your education or work? Explore online courses to continue studying, build professional skills, and connect with experts.' A 'View courses' button is positioned below the text. To the right of the text is a collage of images showing people in a classroom or meeting. At the bottom, there are three circular course collection cards: 'Boredom-busting courses', 'Boost your career courses', and a partially visible one on the right. Each card has a 'View courses' button. The text 'Explore top online courses' is centered at the bottom of the page.

FutureLearn

Subjects ▾ Courses ▾ Using FutureLearn ▾

Search online courses

Keep learning, wherever you are

## Stay connected

Is coronavirus affecting your education or work? Explore online courses to continue studying, build professional skills, and connect with experts.

[View courses](#)

COURSE COLLECTION

Boredom-busting courses

[View courses](#)

COURSE COLLECTION

Boost your career courses

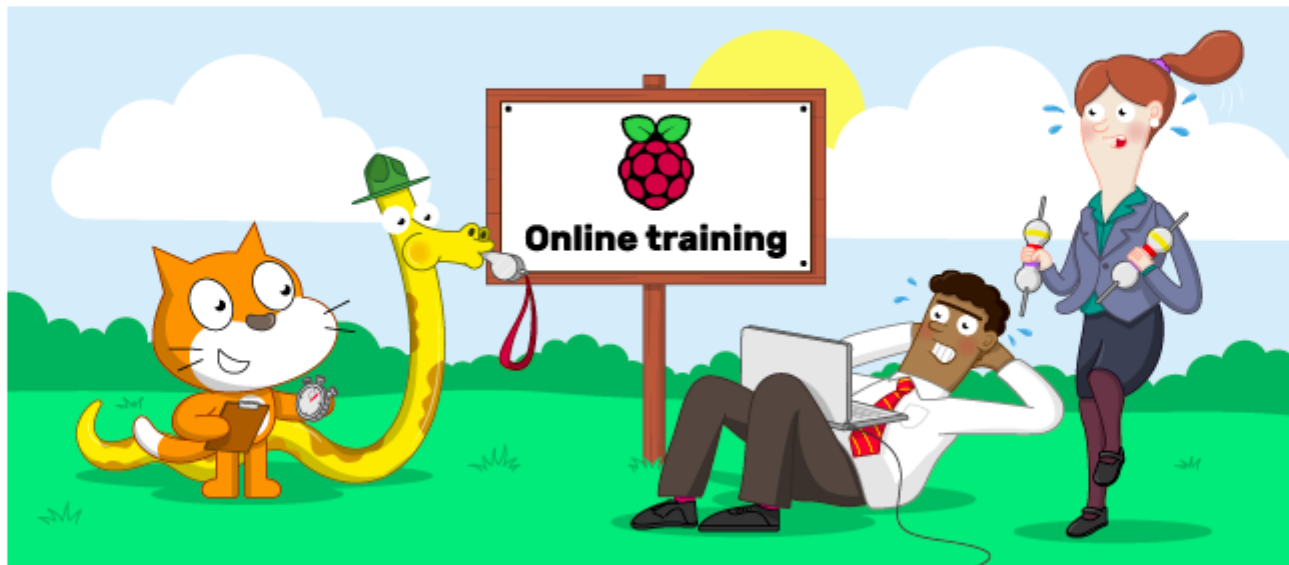
[View courses](#)

Explore top online courses

<https://www.futurelearn.com>



## Raspberry Pi Foundation



## About

26 corsi prodotti da RASPBERRY PI FOUNDATION

# Altre risorse 3



Tra i 26 corsi prodotti da RASPBERRY PI FOUNDATION:

Object-oriented Programming in Python: Create Your Own Adventure Game

Networking with Python: Socket Programming for Communication

**Teaching Physical Computing with Raspberry Pi and Python**

**Programming with GUIs (GUIZERO)**

**Understanding Maths and Logic in Computer Science**

**Representing Data with Images and Sound: Bringing Data to Life (IMAGE)**

Programming 101 ; Programming 102 ; Programming 103

# Altre risorse 4



Dr. Charles R. Severance (MSU)



[Courses](#) / [IT & Computer Science](#)

## Programming for Everybody (Getting Started with Python)

Master the basics of Python programming, and learn how to use programming tools and variables with the University of Michigan.



[Join course for free](#)

[Overview](#) [Topics](#) [Start dates](#) [Requirements](#) [Educators](#) [Try it](#) [More courses](#)



Duration  
6 weeks



Weekly study  
4 hours



Learn  
Free



Extra benefits  
From €59  
[Find out more](#)



Python è un linguaggio di programmazione interpretato, “high-level”, “general-purpose”, e “open-source”.

E’ stato ideato da Guido van Rossum (un informatico olandese) all'inizio degli anni ‘90.

Il nome è legato alla passione di Van Rossum per i Monty Python (un gruppo comico britannico).



Python enfatizza la leggibilità del codice con l'uso della spaziatura.

I suoi costrutti linguistici e l'approccio orientato agli oggetti mirano ad aiutare i programmatori a scrivere codice chiaro e logico per progetti su piccola e grande scala.

Python può essere considerato una versione moderna di Visual Basic: è adatto a sviluppare applicazioni distribuite, per la produzione di scripting, per la computazione numerica e la prototipazione.





Python supporta molteplici paradigmi di programmazione: la programmazione procedurale classica e quella orientata agli oggetti. Python è spesso descritto come un linguaggio "batterie incluse" per la libreria standard completa e le molte librerie aggiuntive disponibili.

E' il linguaggio di programmazione principale su Raspberry PI con il sistema operativo Raspian.

# Python e IDLE



## Per cominciare ...

- Installazione ed uso di Python e IDLE.
- Primi esempi (“Ciao Mondo!”).
- Uso interattivo e tramite script.
- Uso diretto dell’interprete Python.



```
# Python 3: List comprehensions
>>> fruits = ['Banana', 'Apple', 'Lime']
>>> loud_fruits = [fruit.upper() for fruit in
fruits]
>>> print(loud_fruits)
['BANANA', 'APPLE', 'LIME']
```

```
# List and the enumerate function
>>> list(enumerate(fruits))
[(0, 'Banana'), (1, 'Apple'), (2, 'Lime')]
```

## Compound Data Types

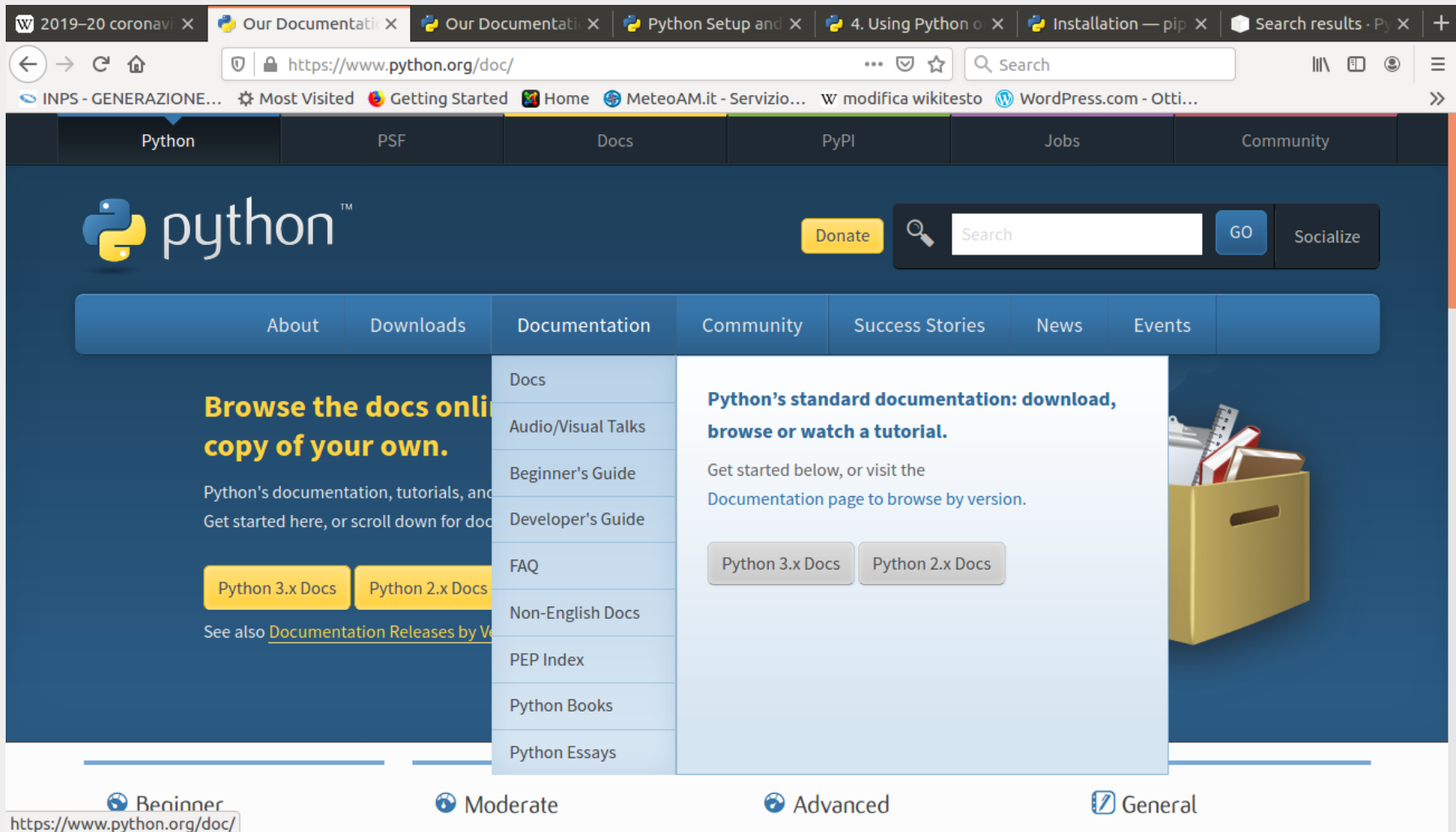
Lists (known as arrays in other languages) are one of the compound data types that Python understands. Lists can be indexed, sliced and manipulated with other built-in functions. [More about lists in Python 3](#)

[1](#)[2](#)[3](#)[4](#)[5](#)

Python is a programming language that lets you work quickly and integrate systems more effectively. [>>> Learn More](#)

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

# La documentazione



The screenshot shows the Python documentation website at <https://www.python.org/doc/>. The browser's address bar and tabs are visible at the top. The website features a dark blue header with the Python logo and navigation links: Python, PSF, Docs, PyPI, Jobs, and Community. Below the header is a secondary navigation bar with links for About, Downloads, Documentation, Community, Success Stories, News, and Events. The main content area is titled "Browse the docs online, or download a copy of your own." and includes buttons for "Python 3.x Docs" and "Python 2.x Docs". A search bar and a "Donate" button are also present. A dropdown menu is open under the "Documentation" link, listing various resources like "Docs", "Audio/Visual Talks", "Beginner's Guide", "Developer's Guide", "FAQ", "Non-English Docs", "PEP Index", "Python Books", and "Python Essays". At the bottom, there are filters for "Beginner", "Moderate", "Advanced", and "General".

<https://www.python.org/doc>



Donate



GO

Socialize

About

Downloads

Documentation

Community

Success Stories

News

Events

## Download the latest source release

Download Python 3.8.2

Looking for Python with a different OS? Python for [Windows](#),  
[Linux/UNIX](#), [Mac OS X](#), [Other](#)

Want to help test development versions of Python? [Prereleases](#),  
[Docker images](#)

Looking for Python 2.7? See below for specific releases



<https://www.python.org/downloads/>

# Interprete python



```
Terminal
Vibe
pi@raspberrypi: ~
stefania-HP-nb stefania 537 > python3
Python 3.5.2 (default, Oct 8 2019, 13:06:37)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
>>>
>>> print("ciao mondo!")
ciao mondo!
>>>
>>>
>>> 1+2+3+4
10
>>>
>>>
>>> 1024*2
2048
>>>
>>>
>>> 2**10
1024
>>>
>>>
>>> 10000/5000
2.0
>>>
>>>
>>> █
```

# Interprete python



```
Text Editor with menu
pi@raspberrypi: ~
stefania-HP-nb PYTHON 546 > ls
00prova.py      8ItalianFlag.py      B_TextEditor_Menu.py      D_matching_emojis.py
0prova.py      8resizableapp.py     C_Tic_or_Tac_2.0.py      E_matching_emojis.py
1MyFirstPyGUI.py  8SwedishFlag.py     C_Tic_or_Tac.py          emojis
2eventgui.py    9align.py            D_button_BW.py           emojis_game_1.0.py
3eventgui.py    A_TextEditor_2.0.py  D_countdown_min_sec.py  emojis_game_2.0.py
4eventgui.py    A_TextEditor_byfellow.py  D_countdown.py          emojis_game.py
5passwd.py      A_TextEditor.py      D_counter_guizero.py    emojis.zip
6heroname.py    beeb.png            D_countup_guizero.py    Infinito.txt
7boxes.py      B_Password_TextBox.py  D_dotty_waffle.py       om.png
7heroname.py    B_TextEditor_Menu_2.0.py  DivinaCommedia.txt     text_file.txt
stefania-HP-nb PYTHON 547 > python3 B_TextEditor_Menu_2.0.py
```

Text Editor with menu

File Options

File Name

Font  Colour  Font Size

```
Sempre caro mi fu quest'ermo colle,
E questa siepe, che da tanta parte
Dell'ultimo orizzonte il guardo esclude.

Ma sedendo e mirando, interminati
Spazi di là da quella, e sovrumani
Silenzi, e profondissima quiete

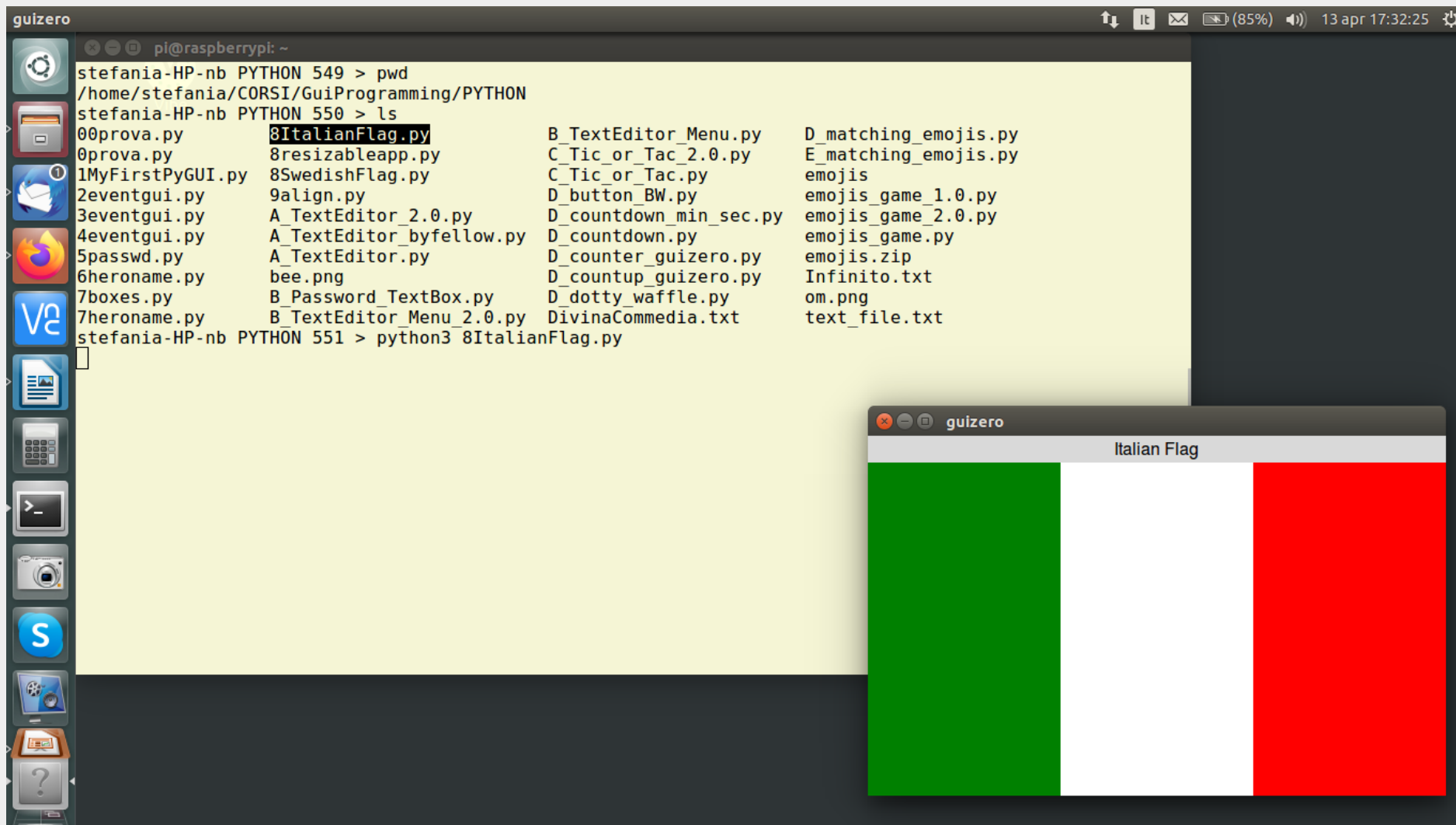
Io nel pensier mi fingo; ove per poco
Il cor non si spaura. E come il vento
Odo stormir tra queste piante, io quello

Infinito silenzio a questa voce
Vo comparando: e mi sovvien l'eterno,
E le morte stagioni, e la presente
```

# Esecuzione di script python

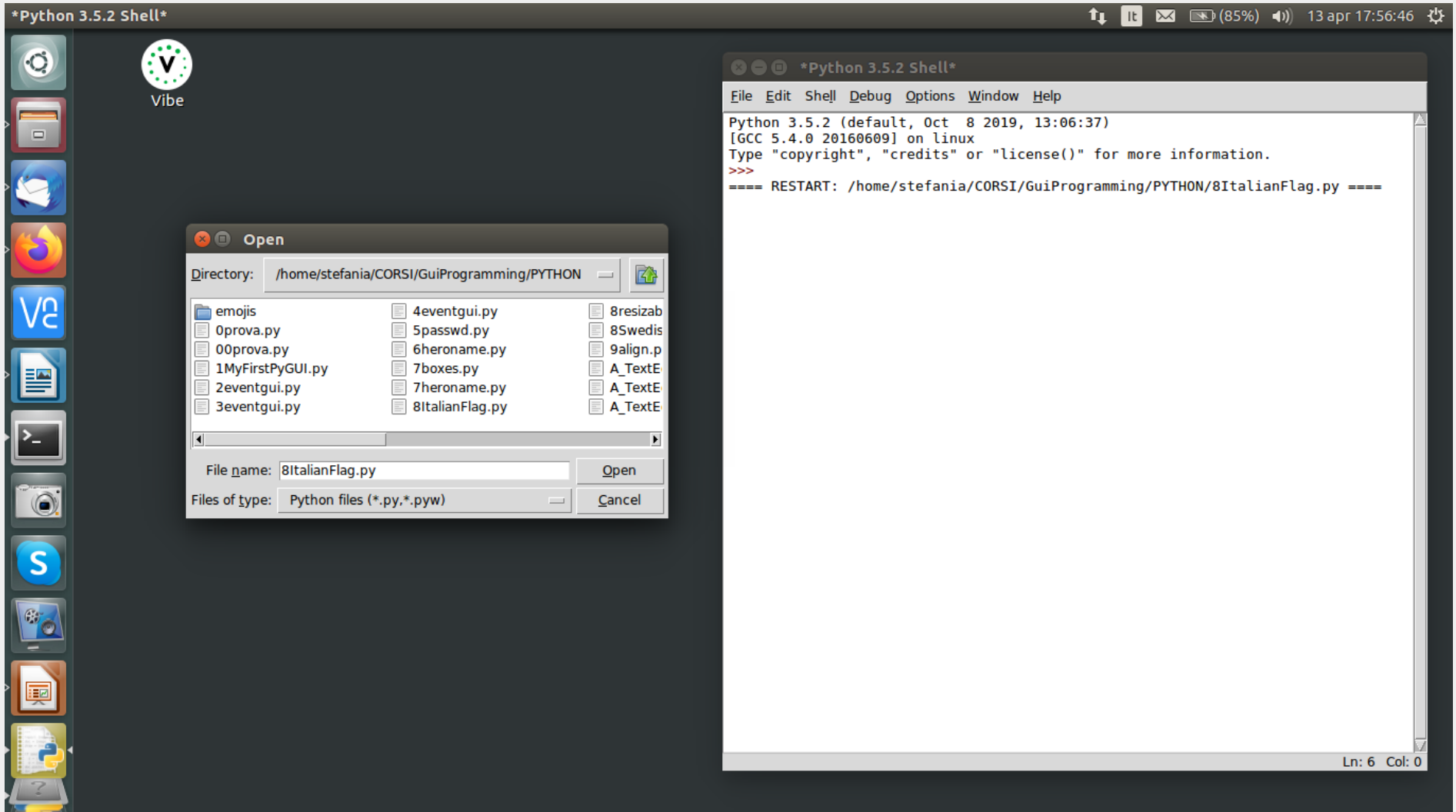


```
guizero pi@raspberrypi: ~
stefania-HP-nb PYTHON 549 > pwd
/home/stefania/CORSI/GuiProgramming/PYTHON
stefania-HP-nb PYTHON 550 > ls
00prova.py      8ItalianFlag.py      B_TextEditor_Menu.py    D_matching_emojis.py
0prova.py      8resizableapp.py     C_Tic_or_Tac_2.0.py     E_matching_emojis.py
1MyFirstPyGUI.py 8SwedishFlag.py     C_Tic_or_Tac.py        emojis
2eventgui.py   9align.py            D_button_BW.py         emojis_game_1.0.py
3eventgui.py   A_TextEditor_2.0.py  D_countdown_min_sec.py emojis_game_2.0.py
4eventgui.py   A_TextEditor_byfellow.py D_countdown.py         emojis_game.py
5passwd.py     A_TextEditor.py      D_counter_guizero.py   emojis.zip
6heroname.py   bee.png              D_countup_guizero.py  Infinito.txt
7boxes.py     B_Password_TextBox.py D_dotty_waffle.py      om.png
7heroname.py   B_TextEditor_Menu_2.0.py DivinaCommedia.txt     text_file.txt
stefania-HP-nb PYTHON 551 > python3 8ItalianFlag.py
█
```





# Usare IDLE 1



The screenshot shows a Linux desktop environment with a dark theme. On the left is a vertical dock with various application icons. The main window is titled "\*Python 3.5.2 Shell\*" and contains a terminal window with the following text:

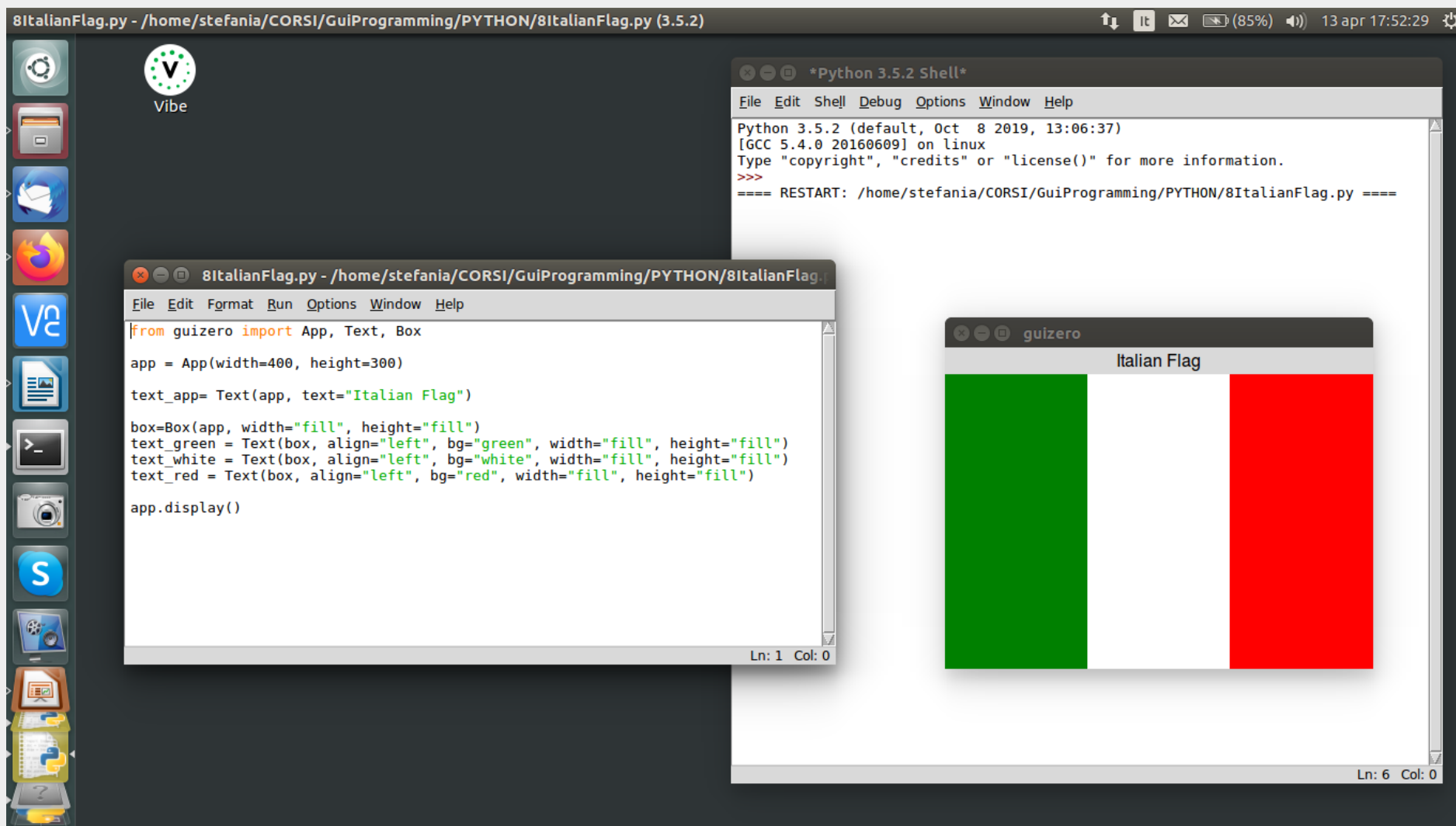
```
Python 3.5.2 (default, Oct 8 2019, 13:06:37)
[GCC 5.4.0 20160609] on linux
Type "copyright", "credits" or "license()" for more information.
>>>
==== RESTART: /home/stefania/CORSI/GuiProgramming/PYTHON/8ItalianFlag.py ====
```

An "Open" file dialog is overlaid on the terminal window. The directory is set to `/home/stefania/CORSI/GuiProgramming/PYTHON`. The file list includes:

emojis	4eventgui.py	8resizab
0prova.py	5passwd.py	8Swedis
00prova.py	6heroname.py	9align.p
1MyFirstPyGUI.py	7boxes.py	A_TextE
2eventgui.py	7heroname.py	A_TextE
3eventgui.py	8ItalianFlag.py	A_TextE

The "File name" field contains `8ItalianFlag.py` and the "Files of type" dropdown is set to "Python files (\*.py;\*.pyw)".

# Usare IDLE 2



The screenshot displays a Linux desktop environment with a sidebar of application icons. The main window is a Python IDE (IDLE) titled "8ItalianFlag.py - /home/stefania/CORSI/GuiProgramming/PYTHON/8ItalianFlag.py (3.5.2)". The code editor shows the following Python code:

```
from guizero import App, Text, Box

app = App(width=400, height=300)

text_app= Text(app, text="Italian Flag")

box=Box(app, width="fill", height="fill")
text_green = Text(box, align="left", bg="green", width="fill", height="fill")
text_white = Text(box, align="left", bg="white", width="fill", height="fill")
text_red = Text(box, align="left", bg="red", width="fill", height="fill")

app.display()
```

The terminal window, titled "\*Python 3.5.2 Shell\*", shows the execution output:

```
Python 3.5.2 (default, Oct 8 2019, 13:06:37)
[GCC 5.4.0 20160609] on linux
Type "copyright", "credits" or "license()" for more information.
>>>
==== RESTART: /home/stefania/CORSI/GuiProgramming/PYTHON/8ItalianFlag.py ====
```

The GUI window, titled "guizero", displays the "Italian Flag" with three vertical bars of green, white, and red. The status bar at the bottom right of the GUI window shows "Ln: 6 Col: 0".

# Perché ?



## Perché imparare a programmare (in python) ?

- è creativo e remunerativo
- può aiutare a risolvere problemi pratici
- ci aiuta a capire come funzionano i dispositivi digitali che usiamo
- è il metodo con cui “addomesticare” computer, tablets e smart-phone  
(vedi scheda 1)



## Vocabolario e grammatica (minima)

- I costrutti base per i programmi procedurali.
- Le variabili e i valori.
- Input/Output.
- Le strutture di controllo.
- Esempi ed esercizi,

# Le Variabili



Con Python le variabili vengono “semplicemente usate”

```
>>> a = 10
>>> print(a)
10
>>> b = (a * 5)/2
>>> print(b)
25
>>>
```

# Le Variabili



Bisogna evitare di usare le parole chiave del linguaggio (33)

and	as	assert	break	class	continue	def
del	elif	else	except	false	finally	for
from	global	if	import	in	is	lambda
none	nonlocal	not	or	pass	raise	return
true	try	while	with	yield		



## Operatori aritmetici (e due operatori fra numeri interi)



Operator Name	Symbol	Try this	Evaluates to
Addition	+	5+2	7
Subtraction	-	5-2	3
Multiplication	*	5*2	10
Real division	/	5/2	2.5
Integer division	//	5//2	2
Modulus	%	5%2	1
Power	**	5**2	25

# Espressioni



Python rispetta il consueto ordine delle operazioni

Operation	Symbols	
Brackets	()	<b>B</b>
Indices	**	<b>I</b>
Division	/ or //	<b>D</b>
Multiplication	*	<b>M</b>
Addition	+	<b>A</b>
Subtraction	-	<b>S</b>



# Operatori di relazione



Operatori di relazione (o confronto), per le espressioni logiche

Usando gli operatori di relazione `>`, `<`, `==`, `!=`, `>=`, `<=`

Si costruiscono espressioni logiche il cui risultato è un valore

Booleano, ovvero determinano una condizione `True` o `False`

Operator name	Symbol	Try this	Evaluates to
Equality	<code>==</code>	<code>5==2</code>	? F
Less than	<code>&lt;</code>	<code>5&lt;2</code>	? F
Greater than	<code>&gt;</code>	<code>5&gt;2</code>	? T
Less than or equal to	<code>&lt;=</code>	<code>5&lt;=2</code>	? F
Greater than or equal to	<code>&gt;=</code>	<code>2&gt;=2</code>	? T
Not equal to	<code>!=</code>	<code>5!=2</code>	? T

# Le stringhe



Le stringhe di caratteri possono essere assegnate a variabili

Inoltre l'operatore + opera con le stringhe una concatenazione:  
ovvero unisce insieme le stringhe:

```
>>> a = 4
>>> b = 6
>>> print(a+b)
10
>>> c = '4'
>>> d = '6'
>>> print(c+d)
46
```

# Stringhe e numeri



Si possono eseguire operazioni mescolando stringhe e numeri

Ad esempio l'operatore `*` può essere usato anche con le stringhe, moltiplicando il contenuto di una stringa per un numero intero. Per esempio:

```
>>> a = 'Prova '
```

```
>>> b = 3
```

```
>>> print(a * b)
```

```
Prova Prova Prova
```

# Input / Output



Per le operazioni di output : *print* ; per input : *input*

Abbiamo già visto in diversi casi che l'istruzione `print` produce l'output su terminale o console del suo argomento in modo sensato.

Per l'operazione di input si usa proprio l'istruzione `input`, che di default legge stringhe:

```
>>> nome = input('Come ti chiami?\n')
```

```
Come ti chiami?
```

```
Gioia
```

```
>>> print(nome)
```

```
Gioia
```

# Commenti



I commenti sono preceduti dal #

I commenti alle istruzioni possono essere posizionati al posto delle istruzioni (a inizio riga) oppure dopo le istruzioni. Si possono inserire righe vuote, a piacere.

# percentuale sull'ora

percentage = (minute \* 100) / 60

Oppure:

percentage = (minute \* 100) / 60 # percentuale sull'ora

# Input numerico



Per le operazioni input numerico occorre la “conversione”

L’istruzione input legge stringhe, da convertire in numeri, al bisogno:

```
>>> prompt = 'Quanti giorni ha il mese di giugno?\n'
```

```
>>> giorni = input(prompt)
```

```
Quanti giorni ha il mese di giugno??
```

```
30
```

```
>>> int(giorni)
```

```
30
```

```
>>> int(giorni) + 1
```

```
31
```

# Controllo semplice: if



## Istruzione condizionale IF-THEN

La forma sintattica della istruzione di controllo IF-THEN è illustrata dal seguente esempio:

```
if (x > 0) :  
    print('x è un numero positivo')
```

Si noti che “then” è tradotto con : e si noti la tabulazione che identifica il blocco.

# Controllo if-then-else



## Istruzione condizionale IF-THEN-ELSE

L'istruzione di controllo IF-THEN-ELSE è illustrata da:

```
if (x > 0) :  
    print('x è un numero positivo')  
else:  
    print('x è un numero negativo o nullo')
```

Si noti che i : sono presenti anche dopo else



# CICLI (while e for)



Le istruzioni per i cicli (loop) con while e for

```
# WHILE
```

```
n=10
```

```
while (n > 0):
```

```
    print(n)
```

```
    n = n - 1
```

```
print('ZERO!')
```

```
# FOR
```

```
n=10
```

```
for i in range(n):
```

```
    print((n-i))
```

```
print('ZERO!')
```



- Input/Output + If-then-else
- Reaction Game + Loop + Statistics
- Reaction Game + Until
- Calcolo dei fogli da ORIGAMI
- Indovina il numero, con suggerimenti

# I mattoni



I mattoni con cui costruire i programmi (procedurali) sono:

INPUT : Get data from the "outside world"

OUTPUT : Display the results of the program

SEQUENTIAL exec : Perform statements one after another.

CONDITIONAL exec : check for conditions and then execute or not. (IF THEN ELSE)

REPEAT : Perform a set of statements repeatedly (LOOPS)

REUSE: Write a set of instructions once and give them a name and then reuse (FUNCTION)



In Python la gestione degli errori usa il costrutto `try-except`

You can think of the `try` and `except` feature in Python as an "insurance policy" on a sequence of statements.

Example:

```
Fahrenheit_Temperature = input('Enter Fahrenheit Temperature:')
```

`try:`

```
    fahrenheit = float(Fahrenheit_Temperature)
```

```
    celsius = (fahrenheit - 32.0) * 5.0 / 9.0
```

```
    print(celsius)
```

`except:`

```
    print('Please enter a number for Fahrenheit Temperature')
```

# Funzioni predefinite



In Python esistono diverse funzioni predefinite (built-in)

type : funzione che ritorna il “tipo” di argomento passato

input : funzione per leggere una stringa (riga) da tastiera

print : funzione che scrive un dato (un risultato) sul video

max e min : ritornano rispettivamente massimo e minimo tra i valori passati

len : ritorna la lunghezza di una stringa (numero di caratteri)

Int: converte l'argomento in numero intero

float: converte l'argomento in numero reale

str: converte l'argomento in stringa

Anche i nomi delle funzioni built-in sono riservati ed è necessario evitare di usarli come nomi di variabili.

# Funzioni di libreria



In Python esistono diverse librerie di funzioni (math, time ...)

*Per usare le funzioni di una libreria occorre “importare” la libreria oppure le singole funzioni di libreria:*

*Esempio per usare sleep da time*

```
import time # poi si usa time.sleep(60)
```

```
from time import sleep # poi si usa sleep(60)
```

*Esempio per usare sqrt da math*

```
import math # poi si usa math.sqrt(2.0)
```

```
from math import sqrt # poi si usa sqrt(2.0)
```

# Definizione di Funzioni



In Python le funzioni sono definite con `def nome_funzione :`

*Esempio di funzione che somma due argomenti*

```
def somma_di_2val( val1, val2) :  
    print(val1+val2)  
    return (val1+val2)
```

(attenzione all'indentazione che definisce il corpo della funzione)

# Parametri e argomenti



Parametri e argomenti sono termini usati per le funzioni

*Gli argomenti sono le variabili che vengono passate alle funzioni*

*math.sqrt(x) è una funzione con 1 argomento e ritorna la radice quadrata di x*

*math.pow(x,y) è una funzione con 2 argomenti e ritorna  $x^y$*

*Dal punto di vista delle funzioni, gli argomenti sono anche indicati come parametri delle funzioni. Il passaggio dei parametri avviene “per valore” - le funzioni NON modificano il valore degli argomenti, ma solo il risultato che eventualmente ritornano*



# Funzioni void e non



Si possono definire funzioni che non ritornano valore (void)

*Le funzioni che non ritornano alcun valore (ad esempio funzioni di puro output) sono le funzioni VOID.*

*Invece le funzioni che eseguono delle operazioni e ritornano un risultato (NON void o fruttuose – fruitful) hanno un tipo: intero, oppure float, oppure bool, oppure stringa.*

*Normalmente il valore di ritorno di una funzione “fruttuosa” viene assegnato ad una variabile, altrimenti va perso e l’esecuzione della funzione è inutile.*

# break e continue



break e/o continue sono direttive per modificare i cicli (loop)

*Esempio di modifica di un ciclo infinito con un break (per terminare il ciclo)*

while True:

```
    line = input('> ')
```

```
    if line == 'done': break
```

```
    print(line)
```

```
print('DONE!')
```

*Esempio di uso del continue (per terminare il solo ciclo corrente), il ciclo NON termina mai*

while True:

```
    line = input('> ')
```

```
    if line[0] == '#': continue # non stampa le righe di solo commento
```

```
    print(line)
```