Python

Loriano Storchi

loriano@storchi.org

http::/www.storchi.org/

Programming Laguanges

- Abbiamo visto che i linguaggi di programmazione possono essere classificati come (non solo ma ...):
  - Dichiarativo
    - Logica
    - Funzionale
  - Imperativo
    - Procedurale
    - Orientato agli oggetti
- Python è imperativo e sia procedurale che orientato agli oggetti

Programming languages

- **Possiamo classificare i linguaggi di programmazione anche come tipizzazione dinamica o statica e tipizzazione forte e debole**
- Tipizzazione debole e dinamica, esempio utilizzando Perl:

```
[redo@banquo tipiz (master)]$ cat test.pl
$i = 2 + "3";
print $i, "\n";
[redo@banquo tipiz (master)]$ perl test.pl
5
[redo@banquo tipiz (master)]$
```

- *Python dynamic and strong typing*

the Python interpreter

- Molto materiale / documentazione disponibile (infatti, questa introduzione è fortemente basata su:
  - http://tdc-www.harvard.edu/Python.pdf
- Python può essere trovato pronto per l'uso sia su Linux che su Mac OS X. Per Windows potete trovare facilmente i binari al seguente URL: http://python.org/
- Molti moduli (librerie) disponibili, nel nostro caso alcuni utili / interessanti: numpy, matplotlib e pystat
- python 2.x vs python 3.x, useremo python 3.x la versione 2.x è stata ora ignorata.

the Python interpreter

```
[redo@virtualinux ~]$ python
Python 2.7.10 (default, Jun 20 2016, 14:45:40)
[GCC 5.3.1 20160406 (Red Hat 5.3.1-6)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> (4*5)/3
6
>>> exit()
[redo@virtualinux ~]$
```

To exit press CTRL-D or type exit()

the Python interpreter

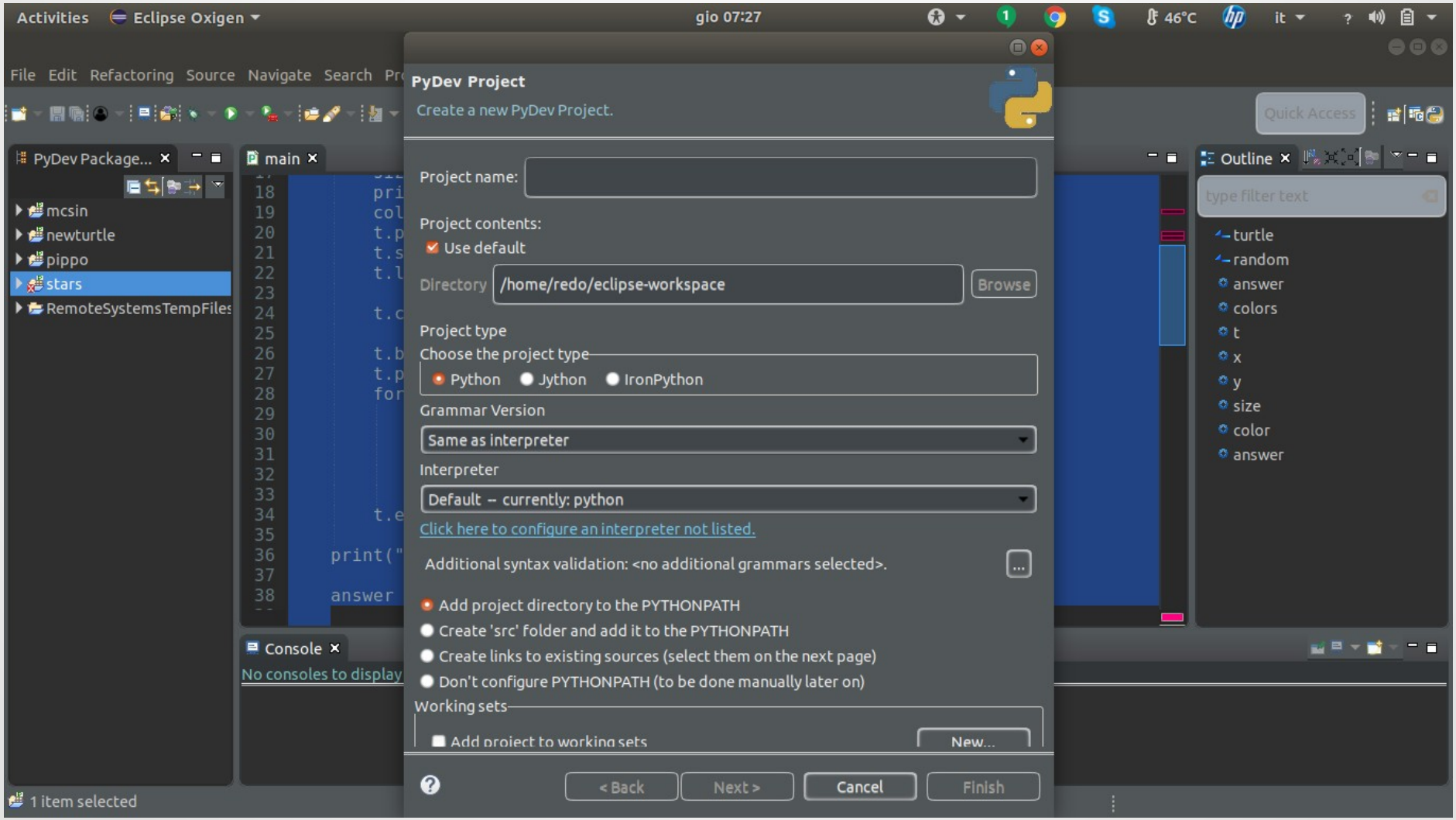
Possiamo semplicemente scrivere il sorgente usando un banale editor ASCII

$ python file.py

```
└─ $ cat file.py
a = 5
b = 5

c = a/b

print(c)

┌redo@buchner /home/redo
└─ $ python3 file.py
1.0
```

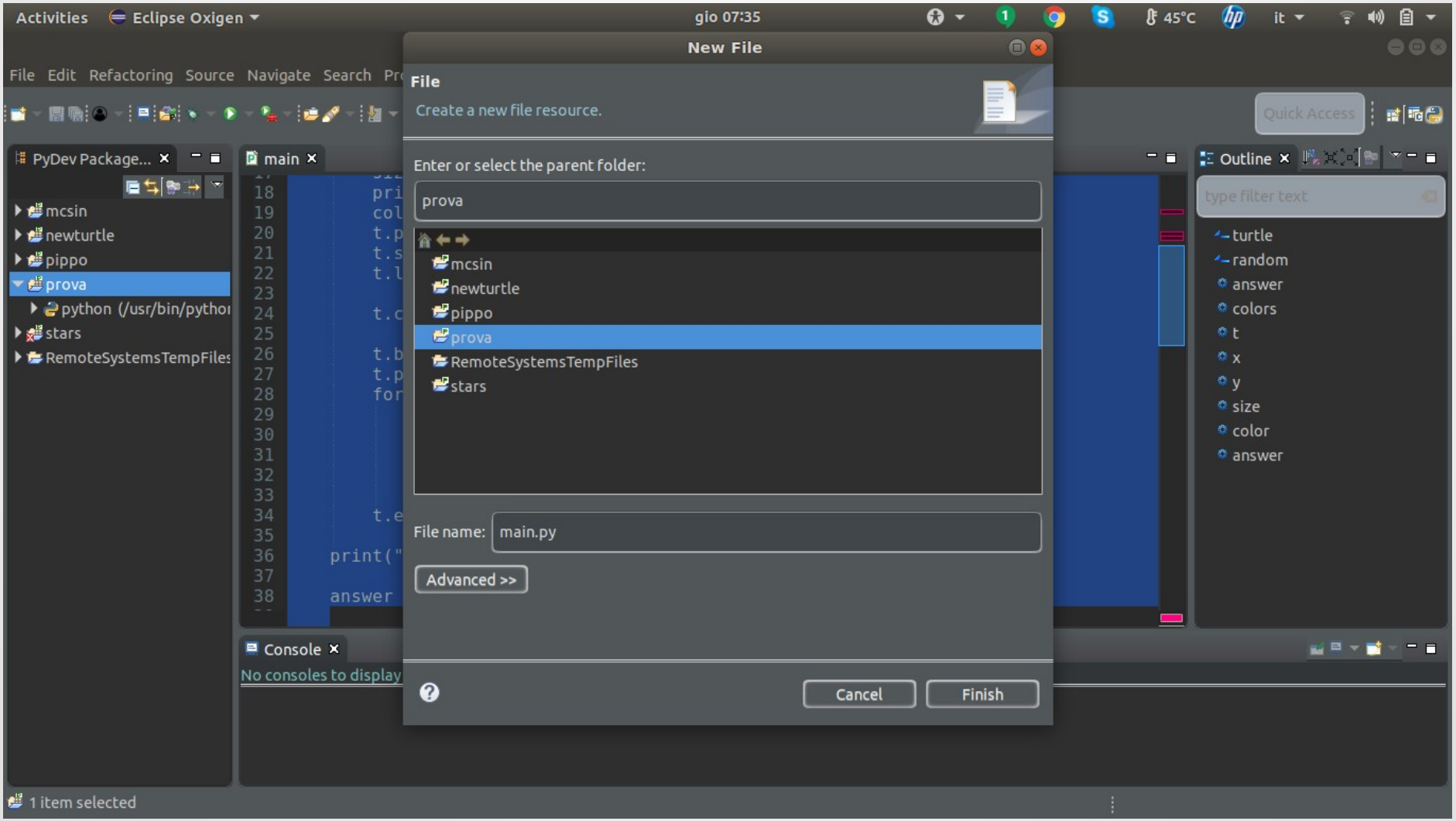
the Python interpreter

Usare un IDE: File → PyDev Project



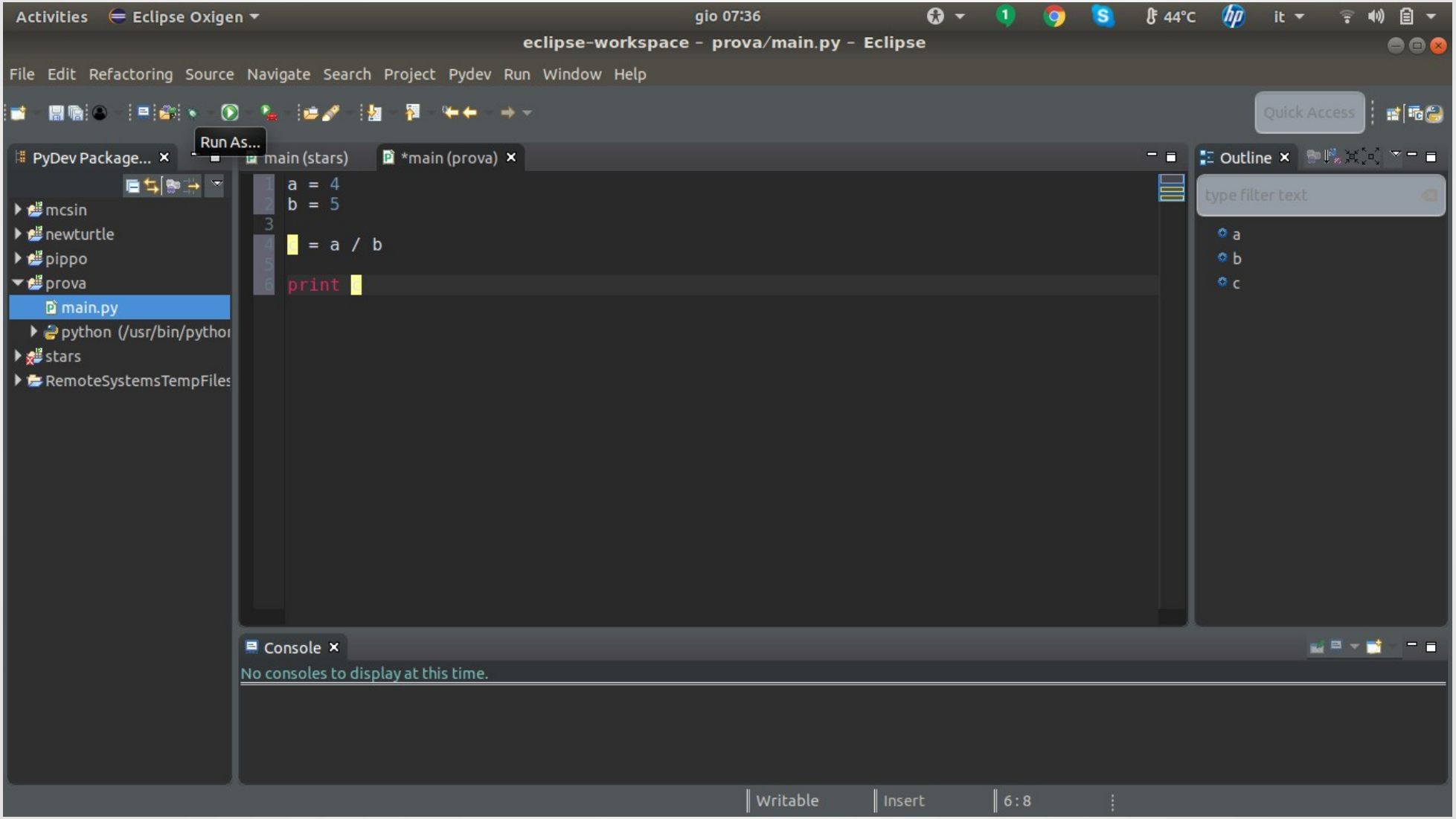
the Python interpreter

Click su Project e poi con il pulsante destro New → File



the Python interpreter

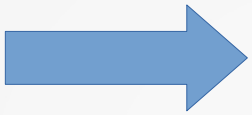
Click Run As

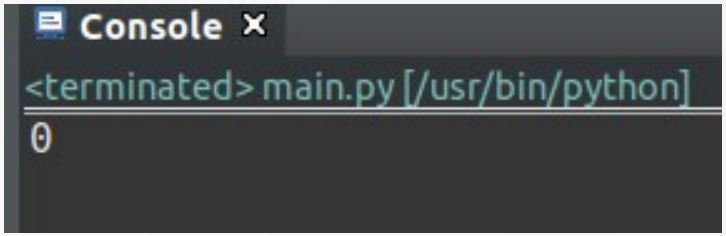


the Python interpreter

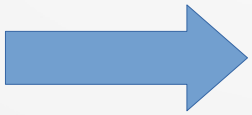
Click Run As

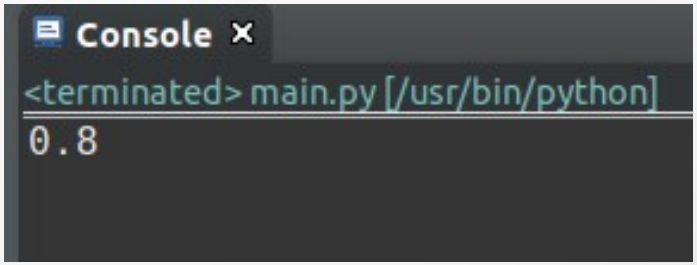
```
1 a = 4
2 b = 5
3
4 c = a / b
5
6 print c
```






```
1 a = 4.0
2 b = 5.0
3
4 c = a / b
5
6 print c
```



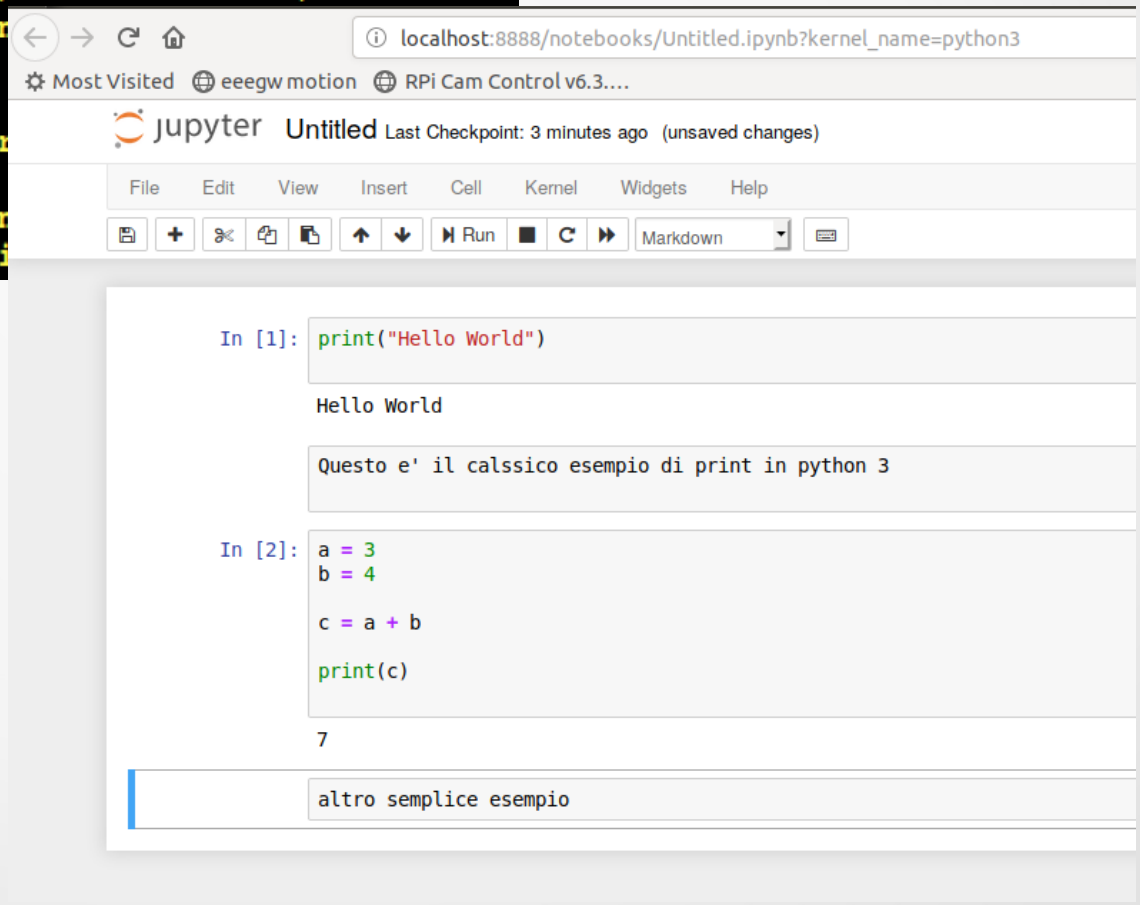


Or Jupyter

Applicazione web in cui puoi creare e condividere documenti che contengono codice live, equazioni, visualizzazioni e testo, Jupyter Notebook è uno degli strumenti ideali

Or Jupyter

```
[redo@buchner FondamentiDiProgrammazione]$ jupyter notebook
[I 18:32:09.367 NotebookApp] Serving notebooks from local dir
mmazione
[I 18:32:09.367 NotebookApp] The Jupyter Notebook is running
[I 18:32:09.367 NotebookApp] http://localhost:8888/?token=510
[I 18:32:09.367 NotebookApp] Use Cor
[C 18:32:09.368 NotebookApp]

    Copy/paste this URL into your br
    to login with a token:
        http://localhost:8888/?toker
[I 18:32:09.617 NotebookApp] Accepti
```



Or Colab

What is Colaboratory?

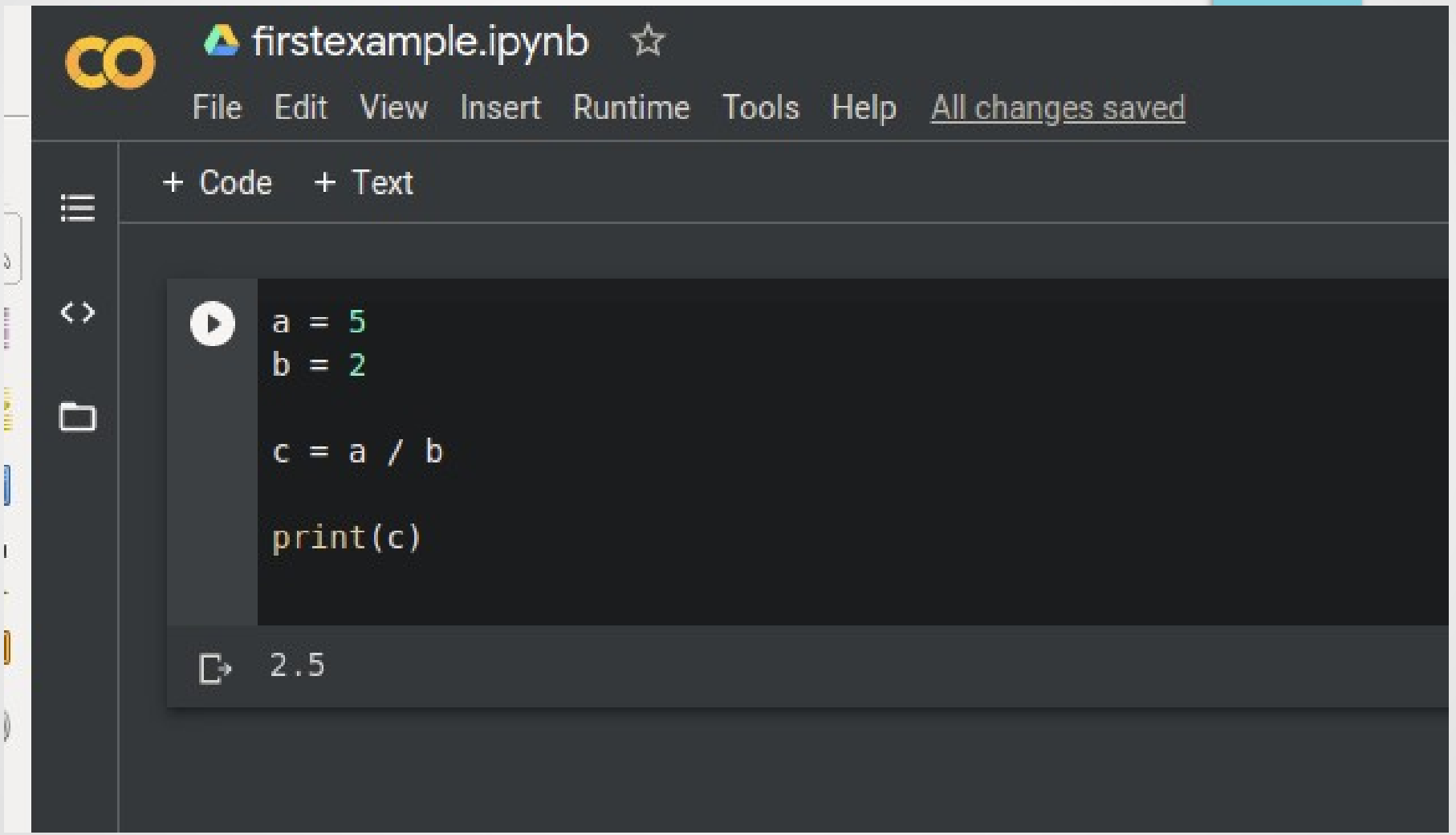
Colaboratory, or "Colab" for short, allows you to write and execute Python in your browser, with

- Zero configuration required
- Free access to GPUs
- Easy sharing

Whether you're a **student**, a **data scientist** or an **AI researcher**, Colab can make your work easier. Watch Introduction to Colab to learn more, or just get started below!

Let’s start →

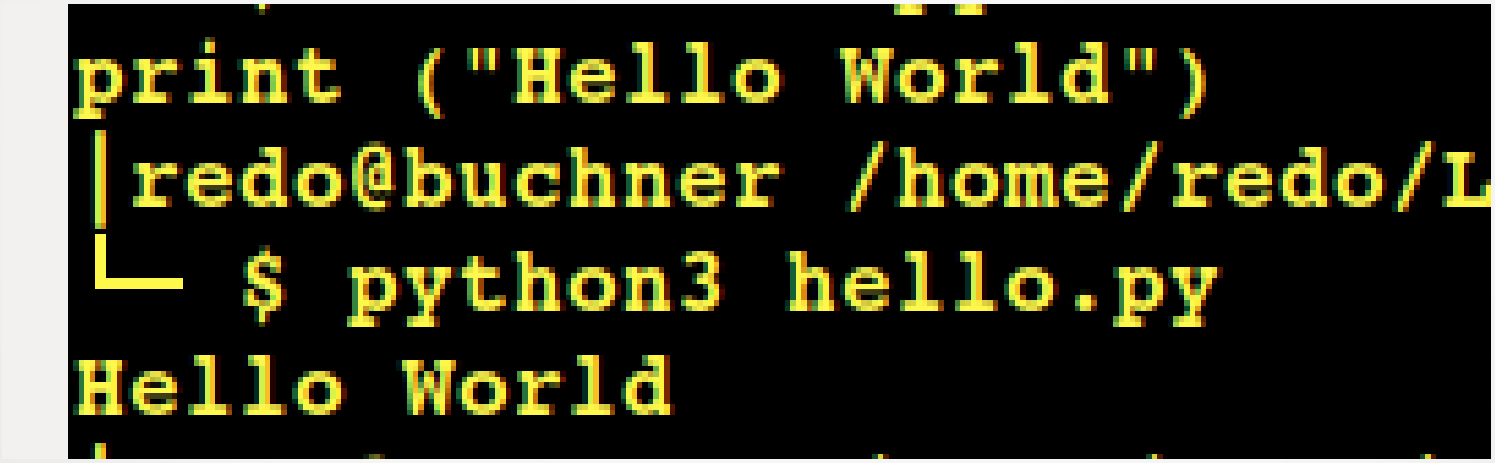
Or Colab


firstexample.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
a = 5
b = 2
c = a / b
print(c)
2.5

HELLO WORLD!

Hello world

- Il classico programma utilizzato per illustrare le basi sintattiche di qualsiasi linguaggio di programmazione



HELLO WORLD 2

Hello world

```
└─ $ cat name.py
name = input("Insert your name: ")

print("Ciao ", name)
```

```
└─ $ python3 name.py
Insert your name: Loriano
Ciao  Loriano
```

Hello world

The variable has no value until the user enters a name.

PYTHON BASICS

Python Basics

- Vediamo un semplice codice che ci permette di illustrare alcune delle caratteristiche di base della sintassi di Python:
  - Posso aggiungere commenti usando il carattere #
  - = viene utilizzato per assegnare valori alle variabili
  - Per fare operazioni tra numeri e variabili posso usare i soliti operatori +, -, *, /

```
x = 34 - 23 # commentare il codice
y = "Hello"
z = 3.45
```

Python Basics

- == è l'operatore che serve a confrontare i valori
- Gli operatori logici sono invece: and, or, not
- L'operatore + può essere utilizzato anche per concatenare le stringhe

```
if z == 3.45 or y == "Hello":
    x = x + 1
    y = y + " World"
```

Python Basics

- print è il comando di base utilizzato per stampare a schermo
- **Le variabili non devono essere dichiarate esplicitamente la prima volta che assegno un valore alla variabile viene creata e gli viene assegnato un tipo**

```
print("Valore di x: ", x)
print("Valore di y: ", y)

print(z)
```

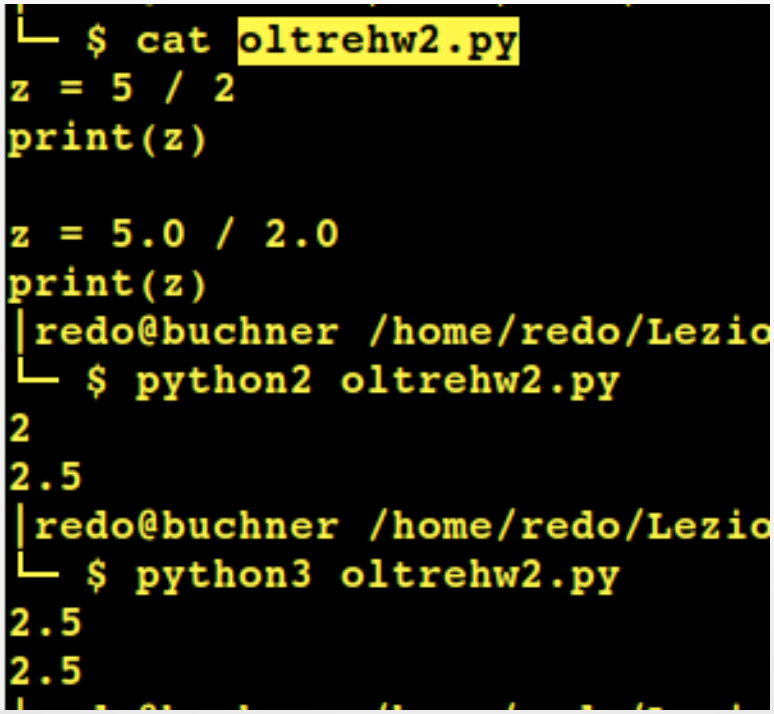
Python Basics

```
x = 34 - 23 # commentare il codice
y = "Hello"
z = 3.45

if z == 3.45 or y == "Hello":
    x = x + 1
    y = y + " World"

print("Valore di x: ", x)
print("Valore di y: ", y)

print(z)
```

```
Valore di x:  12
Valore di y:  Hello World
3.45
```

Python Basics

- Non ci sono caratteri di fine riga, se una riga di codice deve essere interrotta su più righe, usa \
- Python 2 Per impostazione predefinita i numeri sono interi, quindi z = 5/2 darà come risultato 2



IF … THEN … ELSE

If … then … else

```
INSERT NUMBER I
IF I LOWER THEN 0
    PRINT “il numero è minore di zero”
ELSE IF I EQUAL TO 0
    PRINT “Il numero è uguale a zero”
ELSE
    PRINT “Il numero è maggiore di zero”
ENDIF
```

If … then … else

- Per identificare i blocchi di codice **in Python, vengono utilizzati spazi vuoti**, non ad esempio {} come in C / C ++

```
└─ $ cat oltrehw4.py
si = input ("inserisci un numero: " )
i = float(si)

if i < 0 :
    print("numero inferiore a zero")
elif i == 0:
    print("inseiro uguale a zero")
else:
    print("numero maggiore di zero")
```

```
inserisci un numero: -31
numero inferiore a zero
inserisci un numero: 3.0
numero maggiore di zero
```

If … then … else

- Oppure puoi usare eval()

```
si = eval(input ("inserisci un numero: " )) # The expression argument
                                            # is parsed and evaluated as a
                                            # Python expression (technically
                                            # speaking, a condition list)
                                            # using the globals and locals
                                            # dictionaries as global and local
                                            # namespace.

if si < 0 :
    print("numero inferiore a zero")
elif si == 0:
    print("inseiro uguale a zero")
else:
    print("numero maggiore di zero")
```

```
└─ $ python3 oltrehw4.py
inserisci un numero: 10
numero maggiore di zero
inserisci un numero: print("si ", si)
si  10
Traceback (most recent call last):
  File "oltrehw4.py", line 20, in <module>
    if si < 0 :
TypeError: '<' not supported between instances of 'NoneType' and 'int'
```

LOOPS

Loops

SET N TO 0

SET n TO 0

Repeat the following:

a. If n >= 10, terminate the repetition, otherwise.
b. Increment N by n
c. PRINT n

PRINT N

Loops

- Per identificare i blocchi di codice in Python, vengono utilizzati spazi vuoti, non ad esempio {} come in C / C ++
- **Python e' case sensitive**

```
└─ $ less oltrehw3.py
N = 0

for n in range(0,10):
  N = N + n
  print(n)

print("Valore finale: ", N)
```

Loops

```
N = 0

for n in range(0,10):
  N = N + n
  print(n)

print("Valore finale: ", N)
```

```
0
1
2
3
4
5
6
7
8
9
Valore finale:  45
```

EXAMPLE

Example of a numerical procedure

- È possibile trovare un algoritmo per risolvere quasi tutti i problemi, ma non tutti. Ad esempio, calcola le soluzioni di un'equazione del secondo ordine:

```
INPUT  A, B, C
COMPUTE D = (B*B)-(4 * A * C)
IF D >= 0.0
  SOL1 = (-1,0 * B + SQRT(D)) / (2,0 * A)
  SOL2 = (-1,0 * B - SQRT(D)) / (2,0 * A)
  PRINT SOL1 AND SOL2
ELSE
  PRINT “non ci sono soluzioni reali”
```

Example of a numerical procedure

```
└─ $ cat solv.py
import math

a = float(input("insert a:"))
b = float(input("insert b:"))
c = float(input("insert c:"))

print("a = ", a, " b = ", b, " c = ", c)

delta = math.pow(b, 2.0) - (4.0 * a * c)

if (delta >= 0):
  tn = math.sqrt(delta)
  sol1 = ((-1.0 * b) + tn) / (2.0 * a)
  sol2 = ((-1.0 * b) - tn) / (2.0 * a)
  print(sol1, sol2)
else:
  print("No real solutions")
```

Example of a numerical procedure

```
    c = float(c1)
```

Console

```
<terminated> main.py [/usr/bin/python]
insert a:1
insert b:15
insert c:2.5
a =  1.0  b =  15.0  c =  2.5
-0.168560850692 -14.8314391493
```

ESERCIZIO 1

EXERCISE 1

- Scrivi un programma in Python che legga 10 numeri, dopo aver calcolato il valore medio e stampato il risultato

```
[redo@banquo esercizipython (master)]$ python ex1.py
inserisci il numero 1 45
inserisci il numero 2 67
inserisci il numero 3 84
inserisci il numero 4 2
inserisci il numero 5 4
inserisci il numero 6 6
inserisci il numero 7 7
inserisci il numero 8 8
inserisci il numero 9 9.0
inserisci il numero 10 13.0
la somma:  245.0
valore medio:  24.5
[redo@banquo esercizipython (master)]$
```

BREAK

Break

- L'interruzione solitamente nidificata sintatticamente in un ciclo for o while, termina il ciclo più vicino, saltando la clausola else opzionale se il ciclo ne ha una.

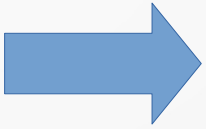
```
└─ $ cat testbreaks.py
for i in range(0,10):
    print("i: ", i)
    if i > 5:
        break;
```



```
i:  0
i:  1
i:  2
i:  3
i:  4
i:  5
i:  6
```

Break and nested loops

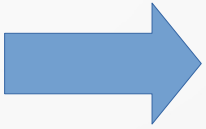
```
└─ $ cat testbreaks1.py
for i in range(10):
    print("loop 1: ", i)
    for j in range(10):
        print("    loop 2: ", j)
        if (j > 5):
            break;
```



```
loop 1:  0
    loop 2:  0
    loop 2:  1
    loop 2:  2
    loop 2:  3
    loop 2:  4
    loop 2:  5
    loop 2:  6
loop 1:  1
    loop 2:  0
    loop 2:  1
    loop 2:  2
    loop 2:  3
    loop 2:  4
    loop 2:  5
    loop 2:  6
loop 1:  2
    loop 2:  0
    loop 2:  1
    loop 2:  2
```

Break and nested loops

```
└─ $ cat testbreaks2.py
for i in range(10):
    print("1) loop 1: ", i)
    for j in range(10):
        print("    loop 2: ", j)
        if (j > 5):
            break;
    print("2) loop 1: ", i)
```



```
1) loop 1:  0
    loop 2:  0
    loop 2:  1
    loop 2:  2
    loop 2:  3
    loop 2:  4
    loop 2:  5
    loop 2:  6
2) loop 1:  0
1) loop 1:  1
    loop 2:  0
    loop 2:  1
    loop 2:  2
    loop 2:  3
    loop 2:  4
    loop 2:  5
    loop 2:  6
2) loop 1:  1
1) loop 1:  2
```

RANDOM NUMBER

Random Number

- https://docs.python.org/2/library/random.html
  - Pseudo Random vs Random
  - cat /dev/urandom
  - cat /dev/urandom | od -vAn -N2 -tu
  - cat /dev/random
  - cat /dev/random | od -vAn -N2 -tu

```
[redo@buchner ~]$ cat /dev/random | od -vAn -N2 -tu2
 38004

^C
[redo@buchner ~]$ cat /dev/random
```

Random Number

- https://docs.python.org/2/library/random.html
  - random.randint(a, b) genera un numero intero random N nell'intevallo a <= N <= b

```
└─ $ cat rnd.py
import random

for i in range(100):
  print(random.randint(0,10))
```

Random Number

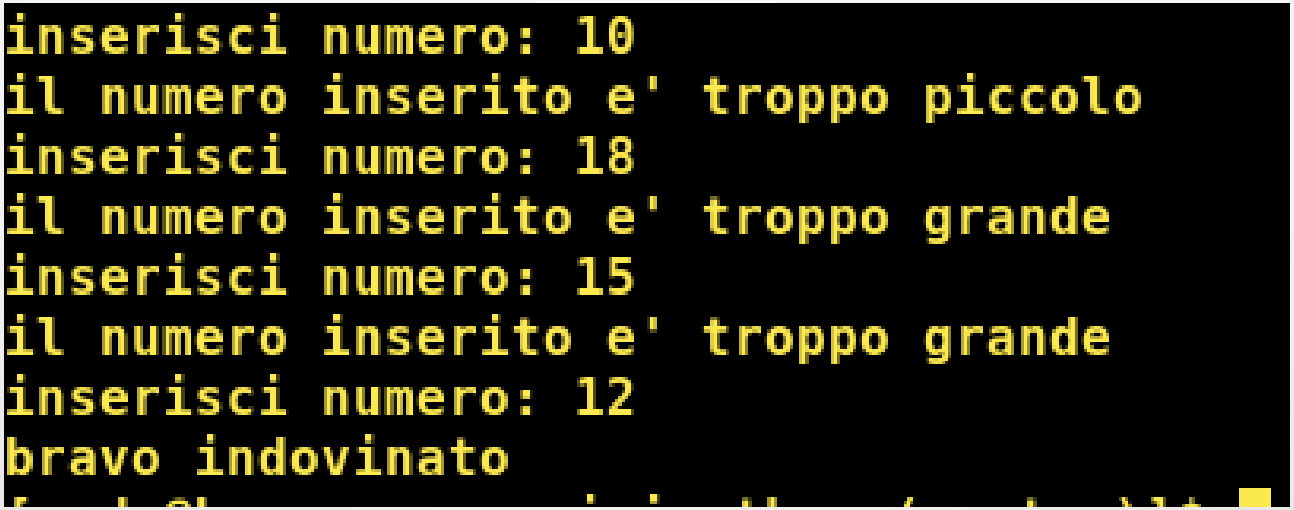
```python
import random

for i in range(100):
  print(random.randint(0,10))
```

```
9
3
3
4
2
4
0
5
4
10
10
2
3
10
```

ESERCIZIO 2

Exercise 2

- Scrivi un programma che generi un numero casuale R compreso tra 0 e 20 e chieda all'utente di indovinare il numero con un massimo di 10 tentativi. Ogni volta che il programma scriverà semplicemente se il numero inserito è maggiore o minore di R. Chiaramente se il numero inserito è uguale al numero casuale R generato il programma termina



Pseudocode

```
GENERATE A RANDOM NUMBER rnd

Repeat the following:
    INPUT b
    IF b IS EQUAL TO rnd
          PRINT “well done”
          BREAK
     ELSE IF b < rnd
          PRINT “inserted number is too small”
     ELSE
          PRINT “inserted number is too big”
    ENDIF
```