Linguistica Computazionale

Laboratorio in Python - III

Alessio Miaschi
ItaliaNLP Lab, Istituto di Linguistica Computazionale (CNR-ILC), Pisa
alessio.miaschi@ilc.cnr.it
https://alemiaschi.github.io/
http://www.italianlp.it/alessio-miaschi/

I moduli (o librerie)

I moduli

- I moduli (o librerie) sono file python contenenti gruppi di funzioni correlate, utilizzati generalmente per operare su dati specifici

- Python include una lista di moduli predefiniti, detti standard (*standard library*)

- È però possibile scaricarne e definirne di nuovi

I moduli

- Per accedere ai contenuti di un modulo, è necessario **importarlo** all'interno del nostro programma, usando il costrutto **import**

```python
import math

def main():
  numero = 25
  radice = math.sqrt(numero)
  print(numero)

main()

# >>> 5.0
```

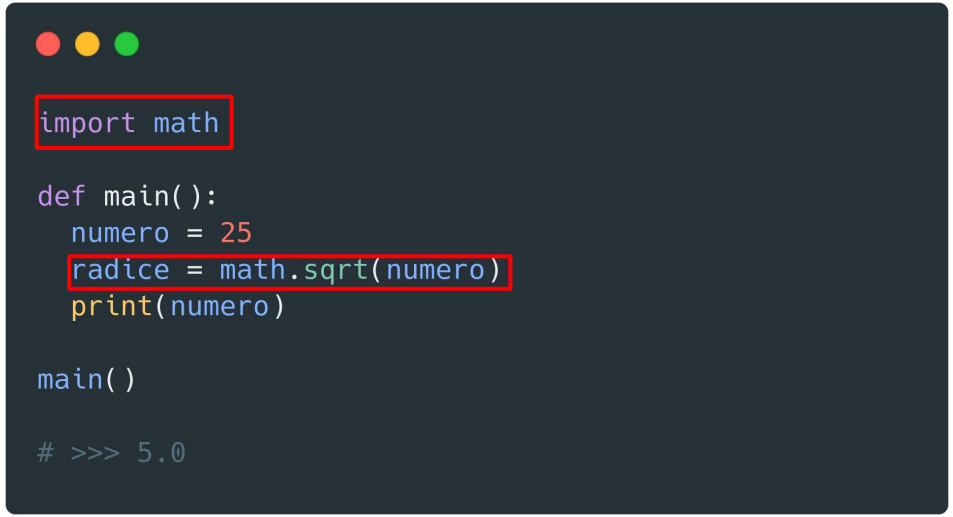
I moduli

- Per accedere ai contenuti di un modulo, è necessario **importarlo** all'interno del nostro programma, usando il costrutto **import**

```python
import math

def main():
  numero = 25
  radice = math.sqrt(numero)
  print(numero)

main()

# >>> 5.0
```

I moduli

- Una volta importato il modulo, sarà possibile accedere alle sue funzioni ricorrendo alla sintassi **<nome_modulo>.<nome_funzione>**



```python
import math

def main():
  numero = 25
  radice = math.sqrt(numero)
  print(numero)

main()

# >>> 5.0
```

I moduli

- È possibile importare anche solo alcune funzioni di un determinato modulo, usando il costrutto **from <nome_modulo> import <nome_funzione>**

```
from math import sqrt

def main():
  numero = 25
  radice = sqrt(numero)
  print(numero)

main()

# >>> 5.0
```

Scaricare nuovi moduli

- È possibile scaricare e installare nuovi moduli ricorrendo a:
  - comando pip (pip3) direttamente da terminale;
  - Tramite le impostazioni di Pycharm

```
alessio@alessio:~$ pip install torch
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: torch in ./.local/lib/python3.6/site-packages (1.7.0)
Requirement already satisfied: dataclasses in /usr/local/lib/python3.6/dist-packages (from torch) (0.6)
Requirement already satisfied: future in ./.local/lib/python3.6/site-packages (from torch) (0.18.2)
Requirement already satisfied: numpy in ./.local/lib/python3.6/site-packages (from torch) (1.19.5)
Requirement already satisfied: typing-extensions in ./.local/lib/python3.6/site-packages (from torch) (3.7.4.3)
```

Scaricare nuovi moduli

- È possibile scaricare e installare nuovi moduli ricorrendo a:
  - comando pip (pip3) direttamente da terminale;
  - Tramite le impostazioni di Pycharm

```
alessio@alessio:~$ pip install torch
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: torch in ./.local/lib/python3.6/site-packages (1.7.0)
Requirement already satisfied: dataclasses in /usr/local/lib/python3.6/dist-packages (from torch) (0.6)
Requirement already satisfied: future in ./.local/lib/python3.6/site-packages (from torch) (0.18.2)
Requirement already satisfied: numpy in ./.local/lib/python3.6/site-packages (from torch) (1.19.5)
Requirement already satisfied: typing-extensions in ./.local/lib/python3.6/site-packages (from torch) (3.7.4.3)
```

Il modulo sys

- Il modulo **sys** ci permette di interagire con il sistema operativo su cui stiamo lavorando

- Uno degli utilizzi principali di questo modulo è quello di passare degli *argomenti* (stringhe, file di testo, ecc) al nostro programma direttamente da riga di comando

Il modulo sys

- Il modulo **sys** ci permette di interagire con il sistema operativo su cui stiamo lavorando

- Uno degli utilizzi principali di questo modulo è quello di passare degli *argomenti* (stringhe, file di testo, ecc) al nostro programma direttamente da riga di comando

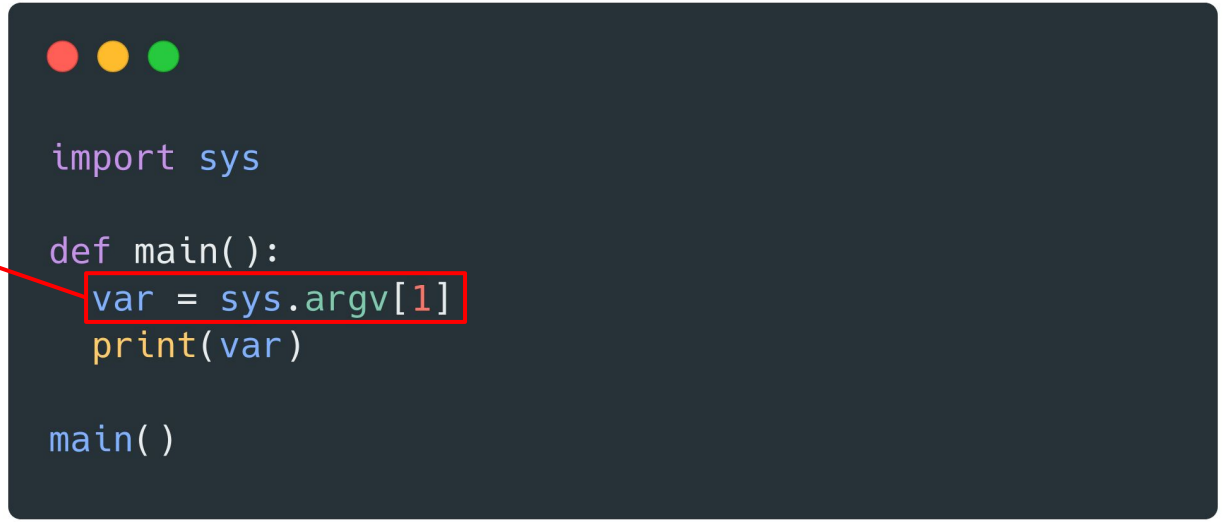
python programma.py argomento_1 argomento_2 argomento_n

Il modulo sys

```
import sys

def main():
  var = sys.argv[1]
  print(var)

main()
```

Il modulo sys

Primo argomento



```
import sys

def main():
  var = sys.argv[1]
  print(var)

main()
```

Il modulo sys

Primo argomento

```python
import sys

def main():
  var = sys.argv[1]
  print(var)

main()
```

```
alessio@alessio:~/Desktop$ python programma.py argomento
argomento
alessio@alessio:~/Desktop$
```

Il modulo sys

```python
import sys

def main():
  num1 = int(sys.argv[1])
  num2 = int(sys.argv[2])
  print(num1+num2)

main()
```

Il modulo sys

```python
import sys

def main():
  num1 = int(sys.argv[1])
  num2 = int(sys.argv[2])
  print(num1+num2)

main()
```

```
alessio@alessio:~/Desktop$ python programma.py 10 20
30
alessio@alessio:~/Desktop$
```

Elaborazione di file di testo

Apertura/chiusura di un file

- Per aprire un file in python si utilizza la funzione **open()**

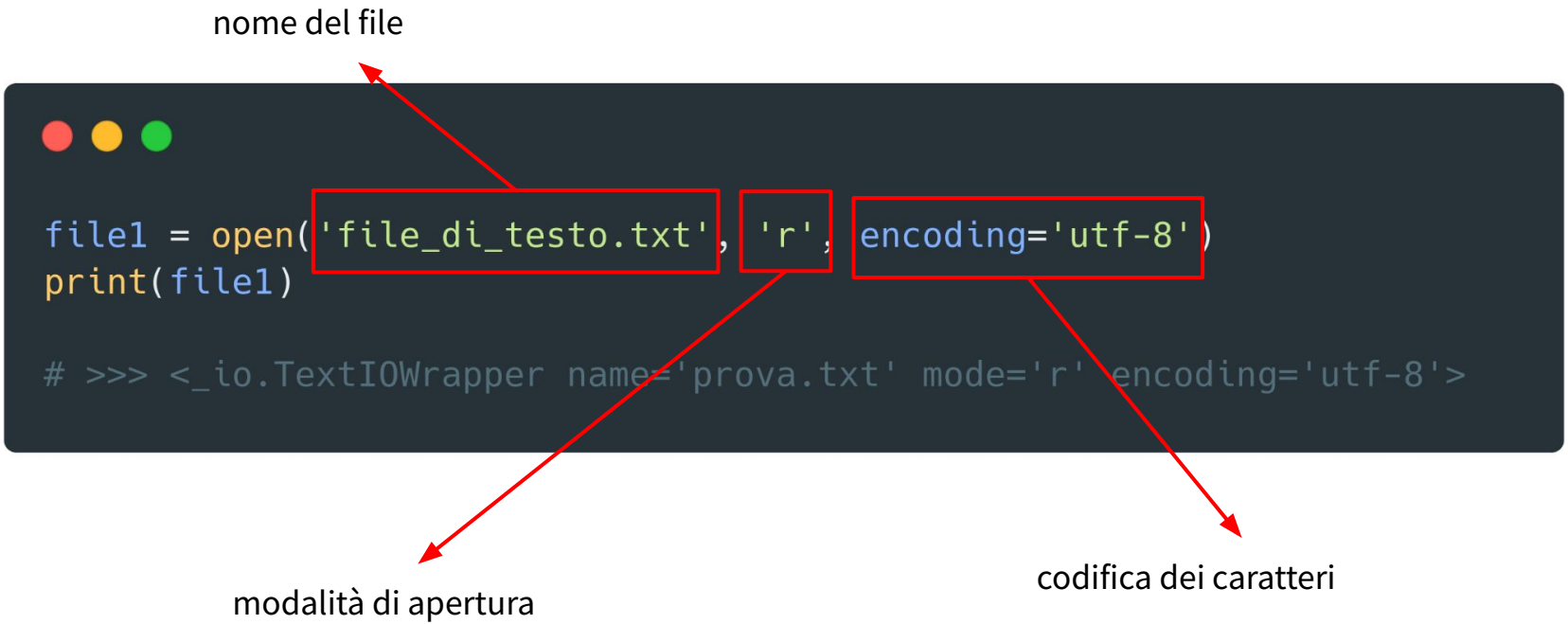
- La funzione **open()** prende in input più argomenti, ma i più importanti sono:
  - **nome del file**: percorso che mi indica la posizione del file
  - **modalità**: ovvero se stiamo aprendo un file in modalità di lettura o di scrittura (**r**, **w** o **a**)
  - **codifica**: la codifica dei caratteri da utilizzare per la lettura del file

- Quando abbiamo finito di eseguire determinate operazione su un file, è sempre consigliato chiuderlo utilizzando il metodo **<nome_file>.close()**

Apertura/chiusura di un file

```
file1 = open('file_di_testo.txt', 'r', encoding='utf-8')
print(file1)

# >>> <_io.TextIOWrapper name='prova.txt' mode='r' encoding='utf-8'>
```

Apertura/chiusura di un file



Apertura/chiusura di un file

```
file1 = open('file_di_testo.txt', 'r', encoding='utf-8')
print(file1)
file1.close()
print(file1)

# >>> <_io.TextIOWrapper name='prova.txt' mode='r' encoding='utf-8'>
# >>> <_io.TextIOWrapper name='prova.txt' mode='r' encoding='utf-8'>
```

Modalità di scrittura

- Utilizzando **w** (*writing*) come modalità di apertura, potremo scrivere all'interno del file di testo

- Nel caso in cui il file non esistesse, l'apertura tramite modalità **w** creerà automaticamente un nuovo file

- **ATTENZIONE**: aprendo un file con la modalità **w** il contenuto precedentemente salvato all'interno del file andrà perso
  - Per poter aggiungere del contenuto, il file dovrà essere aperto con la modalità **a**

Metodi dei file object

Metodo	Descrizione
file.read()	Legge e restituisce l'intero file in una stringa
file.read(*n*)	Legge e restituisce *n* caratteri dei file
file.readline()	Legge e restituisce una riga del file
file.readlines()	Legge e restituisce l'intero file (lista di righe)
file.write(*s*)	Scrive la stringa *s* all'interno del file
file.close()	Chiude il file

Esercizi

- Scrivere un programma che legga un file di testo con la seguente struttura:

```
Pippo 22
Pluto  34
Alessio          29
...
```

  e memorizzi il suo contenuto all'interno di un dizionario (es. {"Pippo": 22, "Pluto": 34, "Alessio": 29})

Esercizi

- Scrivere un programma che legga un file di testo contenente un nome per riga e scriva all'interno di un nuovo file i nomi e la loro rispettiva lunghezza in termini di caratteri. Ad esempio, dato il file:

  Alessio
  Pippo
  Pluto
  Ciao

  il file di output dovrà presentare la seguente struttura:

  Alessio, lunghezza = 7
  Pippo,lunghezza = 5
  Pluto, lunghezza = 5
  Ciao, lunghezza = 4