



Python for Beginners

Modulo IV

Alessio Miaschi

Dipartimento di Informatica, Università di Pisa
ItaliaNLP Lab, Istituto di Linguistica Computazionale (ILC-CNR)

alessio.miaschi@phd.unipi.it
<https://pages.di.unipi.it/miaschi/>

Il modulo *sys*

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

```
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




```
import sys

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

main()
```


Il modulo sys



```
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

nome del file

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

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

modalità di apertura

codifica dei caratteri

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
<code>file.read()</code>	Legge e restituisce l'intero file in una stringa
<code>file.read(<i>n</i>)</code>	Legge e restituisce <i>n</i> caratteri dei file
<code>file.readline()</code>	Legge e restituisce una riga del file
<code>file.readlines()</code>	Legge e restituisce l'intero file (lista di righe)
<code>file.write(<i>s</i>)</code>	Scrive la stringa <i>s</i> all'interno del file
<code>file.close()</code>	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

Esercizi

- Scrivere un programma che prenda in input un file e che restituisca su un nuovo file i vari paragrafi con i nomi propri anonimizzati (es. “Biden è andato [...]” con “NOME_PROPRIO è andato [...]”)
 - Si può assumere che le parole a inizio frase non siano nomi propri!
- Scrivere un programma che prenda in input due file di testo e che restituisca in output la *Type/Token Ratio* (TTR) di entrambi. La TTR è così calcolata:

$$TTR = \frac{V}{T_n}$$

dove V corrisponde al vocabolario (il totale delle parole **diverse** nel testo) e T_n è il totale delle parole nel testo

Dati strutturati e semi-strutturati

Tipologia di dati

- I dati possono essere classificati in:
 - Strutturati;
 - Semi-strutturati.
 - Non strutturati.

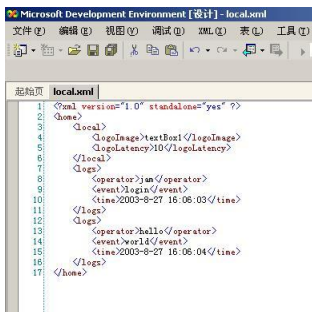
Tipologia di dati

- I dati possono essere classificati in:
 - Strutturati;
 - Semi-strutturati.
 - Non strutturati.

Strutturati

Semi-Strutturati

Non strutturati



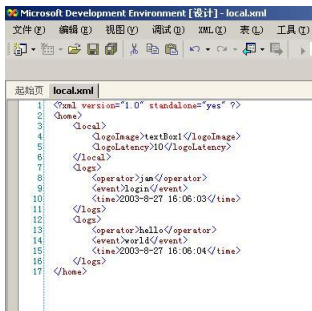
Tipologia di dati

- I dati possono essere classificati in:
 - Strutturati;
 - Semi-strutturati.
 - Non strutturati.

Strutturati

Semi-Strutturati

Non strutturati



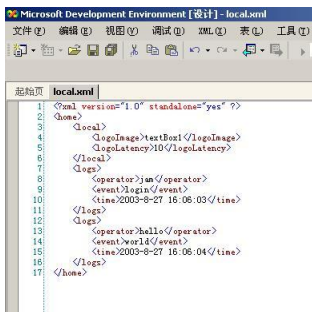
Tipologia di dati

- I dati possono essere classificati in:
 - Strutturati;
 - Semi-strutturati.
 - Non strutturati.

Strutturati

Semi-Strutturati

Non strutturati



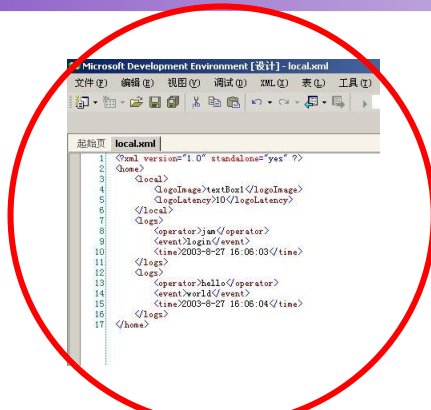
Tipologia di dati

- I dati possono essere classificati in:
 - Strutturati;
 - Semi-strutturati.
 - Non strutturati.

Strutturati

Semi-Strutturati

Non strutturati



Dati strutturati (*csv* e *tsv*)

Dati strutturati

- I dati strutturati sono tutti quei dati che rispettano un set di regole predeterminato
- I dati strutturati dipendono da uno **schema** e possono essere rappresentati con una struttura a “righe x colonne”
- Alcuni esempi:
 - File excel;
 - Csv (*comma-separated values*) e tsv (*tab-separated values*).

File *csv* e *tsv*

- File “tabulati” tra i più diffusi per immagazzinare dati strutturati, generalmente usati per gestire grandi moli di dati
- Le due tipologie di file si distinguono dal carattere con cui vengono separati gli elementi:
 - virgola per i csv;
 - tab (`\t`) per i tsv.

File *csv* e *tsv*

- File “tabulati” tra i più diffusi per immagazzinare dati strutturati, generalmente usati per gestire grandi moli di dati
- Le due tipologie di file si distinguono dal carattere con cui vengono separati gli elementi:
 - virgola per i csv;
 - tab (`\t`) per i tsv.

Nome	Anni	Nazione
Marco	28	Italia
Sarah	32	Inghilterra



CSV

```
"Nome","Anni","Nazione"  
"Marco",28,"Italia"  
"Sarah",32,"Inghilterra"
```

File *csv* e *tsv*

- File “tabulati” tra i più diffusi per immagazzinare dati strutturati, generalmente usati per gestire grandi moli di dati
- Le due tipologie di file si distinguono dal carattere con cui vengono separati gli elementi:
 - virgola per i *csv*;
 - tab (`\t`) per i *tsv*.

Nome	Anni	Nazione
Marco	28	Italia
Sarah	32	Inghilterra



tsv

```
"Nome"      "Anni"      "Nazione"
"Marco"     28          "Italia"
"Sarah"     32          "Inghilterra"
```

File csv e tsv

- In python possiamo leggere/modificare/eseguire operazioni su file csv e tsv:
 - Utilizzando i metodi **.strip()** e **.split()** visti nella lezione precedente
 - Sfruttando le potenzialità di moduli specifici (e.g. *csv*, *pandas*, ecc)

```
file_1 = open('mio_file.csv', 'r', encoding='utf-8')
f = file_1.readlines()
for line in f:
    elementi = line.rstrip('\n').split(',')
    print(elementi)

# >>> ['Nome', 'Anni', 'Nazione']
# >>> ['Marco', '28', 'Italia']
# >>> ['Sarah', '32', 'Inghilterra']
```

File csv e tsv

- In python possiamo leggere/modificare/eseguire operazioni su file csv e tsv:
 - Utilizzando i metodi `.rstrip()` e `.split()` visti nella lezione precedente
 - Sfruttando le potenzialità di moduli specifici (e.g. *csv*, *pandas*, ecc)

```
file_1 = open('mio_file.csv', 'r', encoding='utf-8')
f = file_1.readlines()
for line in f:
    elementi = line.rstrip('\n').split(',')
    print(elementi)

# >>> ['Nome', 'Anni', 'Nazione']
# >>> ['Marco', '28', 'Italia']
# >>> ['Sarah', '32', 'Inghilterra']
```

```
import pandas as pd

df = pd.read_csv('mio_file.csv', delimiter=',')
print(df)

# >>>
#   Nome   Anni  Nazione
# 0 Marco   28   Italia
# 1 Sarah   32  Inghilterra
```


Il modulo pandas

- Il modulo **pandas** fornisce tutta una serie di operazioni per effettuare calcoli/statistiche su file strutturati:
 - Aggiunta/eliminazione colonne/righe/celle;
 - Operazioni aritmetiche su colonne/righe/celle;
 - Correlazioni statistiche;
 - ecc.
- Documentazione del modulo disponibile al seguente link:
https://pandas.pydata.org/docs/user_guide/index.html

Dati semi-strutturati (*JSON* e *XML*)

Dati semi-strutturati

- I dati semi-strutturati sono forme di dati che, sebbene organizzati secondo delle strutture, non sono conformi alla struttura formale dei modelli relazionali e/o tabellari
- Contengono comunque etichette, marcatori, ecc, per poter separare gli elementi e rafforzare le gerarchie all'interno di un dato
- Esempi:
 - JSON (**J**ava**S**cript **O**bject **N**otation);
 - XML (**e**Xtensible **M**arkup **L**anguage);

JavaScript Object Notation (JSON)

```
{
  "Studenti":
  [
    {
      "Nome": "Marco",
      "Cognome": "Rossi",
      "Matricola": 54039,
      "Valutazioni": [21, 22, 30, 28, 27]
    },
    {
      "Nome": "Paola",
      "Cognome": "Bianchi",
      "Matricola": 44432,
      "Valutazioni": [22, 30, 30, 29, 25, 20, 30]
    }
  ]
}
```

JavaScript Object Notation (JSON)

- Tramite il modulo **json** è possibile leggere, creare, modificare file json in python
- In python, i file json vengono trattati come dizionari


```
import sys
import json

def main(json_file):
    data = open(json_file, 'r', encoding='utf-8')
    content = data.read()
    diz = json.loads(content)
    print(diz['Studenti'][0])

main(sys.argv[1])

# >>> {'Nome': 'Marco', 'Cognome': 'Rossi', 'Matricola':
54039, 'Valutazioni': [21, 22, 30, 28, 27]}
```

eXtensible Markup Language (XML)



```
<root>
  <doc id="doc_1">
    <titolo>Titolo 1</titolo>
    <testo>Testo del documento 1...</testo>
  </doc>
  <doc id="doc_2">
    <titolo>Titolo 2</titolo>
    <testo>Testo del documento 2...</testo>
  </doc>
</root>
```

eXtensible Markup Language (XML)

- In python esistono diversi moduli per l'interfacciamento con i file XML
- Il modulo *xml.etree.ElementTree* permette di:
 - navigare all'interno di un file XML;
 - modificare la struttura di un file XML;
 - crearne di nuovi.

eXtensible Markup Language (XML)

- In python esistono diversi moduli per l'interfacciamento con i file XML
- Il modulo *xml.etree.ElementTree* permette di:
 - navigare all'interno di un file XML;
 - modificare la struttura di un file XML;
 - crearne di nuovi.

```
import sys
import xml.etree.ElementTree as ET

def main(doc):
    tree = ET.parse(doc)
    root = tree.getroot()
    for doc in root.iter('doc'):
        print(doc.get('id'))
        print(doc.find('titolo').text)
        print(doc.find('testo').text)

main(sys.argv[1])

# >>> doc_1
# >>> Titolo 1
# >>> Testo del documento 1...
# >>> doc_2
# >>> Titolo 2
# >>> Testo del documento 2 ...
```


Riferimenti

Imparare e approfondire python:

- Learning Python (https://cfm.ehu.es/ricardo/docs/python/Learning_Python.pdf)
- Python ABC (<https://pythonitalia.github.io/python-abc/>)

Python per l'elaborazione automatica del linguaggio:

- Natural Language Processing with Python
(<http://www.datascienceassn.org/sites/default/files/Natural%20Language%20Processing%20with%20Python.pdf>)
- **NLTK**: libreria specifica per l'elaborazione del linguaggio naturale (<https://www.nltk.org/>)