Laboratorio Statistica e Analisi Dati
Lezione 1 - Introduzione e Strutture Per Analisi Dati

Luca Cermenati

2017/2018

Programma

1. Introduzione, Script in Python, Tipi Predefiniti, Serie
2. Data Frame
3. Strutture di controllo del flusso, funzioni
4. Grafici
5. Variabili Aleatorie
6. Esercizi di riepilogo
7. Calcolo di probabilità
8. Calcolo di probabilità
9. Risoluzione di un tema di esame
10. Simulazione di un tema di esame

Installare Python

Mac OS X 10.2 e succesivi includono una versione pre-installata di Python.
Nei sistemi Linux spesso Python è già presente nel setup di base.

Per verificare se Python è già installato è sufficiente aprire una finestra del terminale e digitare il comando python

```
MacBook-Pro-di-Luca:~ lucacermenati$ python
Python 2.7.10 (default, Feb  7 2017, 00:08:15)
[GCC 4.2.1 Compatible Apple LLVM 8.0.0 (clang-800.0.34)] on darwin
Type "help", "copyright", "credits" or "license" for more
information.
>>>
```

Altrimenti è necessario installare Python scaricando dal sito https://www.python.it la versione adatta. La versione che verrà utilizzata nel corso è la 2.7

Interprete interattivo

Il comando python apre l'interprete interattivo.
L'interprete valuta le istruzioni python inserite a riga di comando e fornisce una risposta.

```
Python 2.7.10 (default, Feb  7 2017, 00:08:15)
[GCC 4.2.1 Compatible Apple LLVM 8.0.0 (clang-800.0.34)]
on darwin
Type "help", "copyright", "credits" or "license" for
more information.
>>>
>>> print('Hello World')
Hello World
>>>
```

La funzione exit( ) è digitata per uscire dall'ambiente interprete

```
>>> print('Hello World')
Hello World
>>>
>>>
>>> exit()
MacBook-Pro-di-Luca:~ lucacermenati$
```

Script

Non è obbligatorio usare l'interprete interattivo (Anzi).
É possibile lanciare un file di testo contenente un programma python (file.py), dalla cartella in cui si trova il file.py che si vuole eseguire col comando python <nomefile.py>

```
MacBook-Pro-di-Luca:~ lucacermenati$
MacBook-Pro-di-Luca:~ lucacermenati$ python prova.py
Sto eseguendo uno script
MacBook-Pro-di-Luca:~ lucacermenati$
```

Variabili in Python

Python è dinamicamente tipato e fortemente tipato

Una variabile inizia ad esistere nel momento in cui le viene assegnato un valore, che ne determina il tipo.

```
>>> a=10000
>>> type(a)
<type ‘int’>

>>> len(a)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: object of type 'int' has no len()
>>>
```

Riferirsi ad una variabile inesistente (a cui non è ancora stato assegnato un valore) genera un errore.

```
>>> b
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'b' is not defined
>>>
```

Funzione Help

Dall'interno dell'ambiente interattivo è possibile aprire la documentazione a riguardo di un particolare oggetto, metodo o attributo semplicemente passando alla funzione help(<nome>) il nome dell'oggetto, … che stiamo cercando.

```
>>> help(str)

Help on class str in module __builtin__:

class str(basestring)
 |  str(object='') -> string
 |
 |  Return a nice string representation of the object.
 |  If the argument is a string, the return value is the same
object.
 |
 |  Method resolution order:
 |      str
 |      basestring
```

Si esce dal manuale interattivo premendo q

Tipi Predefiniti in Python

1. Stringhe ‘questa è una stringa’ “questa pure”

2. Dizionari { ‘chiave’: valore } {1: ‘valore’}

3. Tuple (1,2,3,’ciao’, (‘a’,’b’,’c’))

4. Liste [ ]

Array

Il modulo numpy mette a disposizione una struttura dati creata appositamente per il calcolo scientifico.
Un Array è un contenitore di dati omogenei, cioè dello stesso tipo. Se si tenta di costruire un array con dati di tipo diverso, questi vengono "trasformati" nel più piccolo tipo adatto a contenerli tutti.

```
>>> numpy.array( [1,4,6.8,10] )
array([  1. ,   4. ,   6.8,  10. ])
```

Creazione di Array

numpy.arange(x): restituisce un array contenente gli interi da 0 a X-1

```
>>> numpy.arange(5)
[0,1,2,3,4]
```

numpy.arange(start, stop, step): permette di scegliere gli estremi della sequenza contenuta nell'array e il passo tra un numero ed il successivo.

```
>>> numpy.arange(2,6,2)
[2,4]
>>> numpy.arange(2,-3,-1)
[2,1,0,-1,-2]
```

numpy.linspace(start, stop, n): permette di scegliere gli estremi e quanti numeri devono comparire all'interno della sequenza

```
>>> numpy.linspace(0,1,4)
[0,  0.3333333,  0.66666667,  1]
```

numpy.ones(tupla contenente dim): restuituisce un array di 1, date le dimensioni.

```
>>> numpy.ones( (1,5) )
[1. , 1. , 1. , 1. , 1.]
```

numpy.zeros(tupla contenente dim): restuituisce un array di 1, date le dimensioni.

```
>>> numpy.zeros( (2,5) )
[[0. , 0. , 0. , 0. , 0.] , [0. , 0. , 0. , 0. , 0.]]
```

Creazione random di Array

numpy.random.randint(low, high, n): restituisce un array di n elementi interi scelti casualmente tra i valori low e high

numpy.random.random(n): restituisce un array di n elementi float scelti casualmente nell'intervallo tra 0.0 e 1.0

https://docs.scipy.org/doc/numpy/reference/routines.random.html

Operazioni con Array

Applicando un operazione logica ad un array si ottiene come risposta un array di tipo booleano contenente nella posizione i-esima il risultato dell'espressione logica valutata per l'elemento i-esimo dell'array.

```
>>> numpy.arange(5) > 3
array([False, False, False, False,  True], dtype=bool)
```

Le operazioni + - * / % ,.... Tra due array vengono eseguite elemento per elemento. Gli array devono avere le stesse dimensioni.

```
numpy.arange(5) + numpy.linspace(0,2,5)
>>> array([ 0. ,  1.5,  3. ,  4.5,  6. ])

>>> numpy.arange(5) * 2
array([0, 2, 4, 6, 8])
```

Selezioni di un sottoinsieme di elementi

Ad un array è possibile accedere in tutti i modi con cui è possibile accedere ad una lista, inoltre

Ottenere una tupla di elementi

```
>>> a[2], a[-1], a[3:6]
(2, 9, array([3, 4, 5]))
```

Slicing tipo [start:stop:step]

```
>>> a[2:9:3]
array([2, 5, 8])

>>> a[::2]
array([0, 2, 4, 6, 8])
```

Accesso con maschere booleane (filtri)

Specificando tra [ ] un array di tipo booleano, selezioniamo dall'array gli elementi la cui posizione corrisponde ad un True nell'array di tipo booleano. Che prende il nome di maschera.

```
>>> a = numpy.arange(10)
>>> mask = a%2 == 0
>>> mask
array([ True, False,  True, False,  True, False,  True, False,
True, False], dtype=bool)
>>>
>>> a[mask]
array([0, 2, 4, 6, 8])
```

in maniera più compatta

```
>>> a[a%2 == 0]
array([0, 2, 4, 6, 8])
```

Serie

Una Serie è un array monodimensionale ed etichettato.
Rappresenta una serie di osservazioni di un certo carattere, fatto su un insieme di individui.
É una struttura dati appartenente al modulo python Pandas.

```
a    0.0
b    1.0
c    2.0
d    3.0
e    4.0
f    5.0
g    6.0
h    7.0

Name: serie_di_esempio, dtype: float64
>>>
```

pandas.Series (data, index, dtype, name, …)

data: array-like, dizionario, contiene i dati che si vuole inserire nella serie.
index: array-like, della stessa lunghezza di data, contiene le etichette.
dtype: permette di specificare il tipo che i dati devono assumere
name: permette di dare un nome alla Serie

É possibile specificare altri parametri, ma anche ometterne alcuni. E ottenere Serie da altre strutture dati.
https://pandas.pydata.org/pandas-docs/stable/generated/pandas.Series.html

```
>>> import pandas
>>> import numpy
>>> oss=numpy.arange(8)
>>> etic=['a','b','c','d','e','f','g','h']
>>> pandas.Series(data=oss, index=etic, name='serie_di_esempio',
dtype=float)
a    0.0
b    1.0
c    2.0
d    3.0
e    4.0
f    5.0
g    6.0
h    7.0
Name: serie_di_esempio, dtype: float64
```

Accesso ai dati di una Serie

Specificando un valore per l'etichetta, o una lista di valori
Specificando un valore posizionale, o una lista di valori

```
>>> s['d']
3.0
>>> s.loc[['h','a']]
h    7.0
a    0.0
Name: serie_di_esempio
```

```
>>> s[[5,6]]
f    5.0
g    6.0
Name: serie_di_esempio
>>> s.iloc[0]
0.0
```

Stile list slicing

```
>>> s[3:6]
d    3.0
e    4.0
f    5.0
Name: serie_di_esempio, dtype: float64
>>>
>>> s[-3:]
f    5.0
g    6.0
h    7.0
Name: serie_di_esempio, dtype: float64
```

Accesso ai dati di una Serie

Come per gli array, applicando un espressione logica ad una Serie si ottiene una serie di Booleani, che può essere utilizzata come maschera per accedere ai soli dati che corrispondo al valore True.

```
>>> s % 2 == 0
a     True
b    False
c     True
d    False
e     True
f    False
g     True
h    False
Name: serie_di_esempio, dtype: bool

>>> s[s % 2 == 0]
a    0.0
c    2.0
e    4.0
g    6.0
```

Metodi della classe Series

Series.min(), Series.max() restituiscono rispettivamente il massimo ed il minimo tra i valori osservati

Series.sum(): restituisce la somma dei valori osservati

Series.sort_values(): restituisce una copia della serie ordinata per valore

Series.sort_index(): restituisce una copia della serie ordinata per etichette

Series.sample(n): restituisce una serie di n elementi scelti uniformemente a caso all'interno di quelli della serie

Series.append(s1,s2,...): concatena una o più serie a quella di partenza

Series.count(): restituisce il numero dei valori osservati che non siano il valore NaN

Ulteriori metodi, attributi e raffinamenti
https://pandas.pydata.org/pandas-docs/stable/generated/pandas.Series.html

Il valore nullo Nan (None)

Il valore Nan rappresenta, in una serie, un dato di cui non è stato possibile fare un'osservazione. Un dato mancante.
É un vero e proprio oggetto del modulo numpy.
Somme, differenze, moltiplicazioni, … con un valore Nan hanno come risultato il valore Nan.

```
>>> oss = range(5)
>>> oss.append(None)
>>> pandas.Series(oss)
0    0.0
1    1.0
2    2.0
3    3.0
4    4.0
5    NaN
```

```
>>> sum(pandas.Series(oss))
nan
>>> pandas.Series(oss).sum()
10.0
>>>
```

Esercizi

0. Lanciare uno script che stampi il vostro nome e cognome.

1. Creare un array i cui elementi siano gli interi che compongono il vostro numero di matricola.
   Sia X la lunghezza dell'array del punto 1.
2. Creare un array di X elementi equispaziati l'uno dall'altro di 1.5
3. Creare un array di X elementi equispaziati nell'intervallo tra 1 e 3
4. Ottenere gli array dati dalla somma tra l'array del punto 1 e quello del punto 2, e quello dato dalla moltiplicazione tra punto 2 e punto 3.
5. Ottenere per ciascuno dei due array del punto 4, un array maschera che in corrispondenza dei numeri divisibile per 3 abbia True, False altrimenti.
6. Ottenere le Serie corrispondenti ai 3 array di partenza.
7. Concatenare le tre Serie e visualizzare il risultato.

Esercizi

Creare la Serie gol in modo che le etichette siano numeri dall'1 al 23. I dati raccolti siano numeri interi tra 0 e 30 scelti casualmente.
La Serie rappresenta i gol segnati da ciascun giocatore di una società calcistica nella stagione appena passata. L'etichetta corrisponde al numero di maglia del giocatore.

Ottenere:
I gol segnati dal capocannoniere
I primi 5 cannonieri, e gli ultimi 5
Chi non ha mai fatto gol durante l'anno
La statistica per i soli titolari, numeri dall'1 all'11, e per le sole riserve
La statistica per i soli attaccanti (numeri dal'9 all'11)
I gol segnati dai giocatori coi numeri di maglia 5,9,10,14 (somma)
Chi ha segnato più di 10 gol e chi ha segnato esattamente 10 gol
Riordinare la serie in ordine di gol effettuati
Riordinare la serie in ordine inverso di numero di maglia