

Tutto quello che avreste voluto sapere su Python (ma non avete mai osato chiedere)

- **Sintassi e funzioni di python:** sintassi particolare e operatori nativi del linguaggio Python

Segnatura	Descrizione
<code>**</code>	Operatore di elevamento a potenza.
<code>/</code>	Operatore di divisione reale.
<code>//</code>	Operatore di divisione intera.
<code><elemento> in <lista/tupla/set/dict></code>	Operatore booleano di appartenenza.
<code><lista>.insert(posizione, elemento)</code>	Inserisce l'elemento in 'posizione' nella lista.
<code><lista>.sort(reverse=False)</code>	Ordina la collezione in base al valore degli elementi.
<code>del <lista/dict>[posizione]</code>	Elimina l'elemento in 'posizione' dalla collezione.
<code>lambda <param> : <result></code>	Crea una funzione anonima che applica l'operazione 'result' ai suoi parametri 'param'.
<code>len(collezione)</code>	Restituisce il numero di elementi di 'collezione'.
<code>sum(<lista/serie/...>)</code>	Restituisce la somma di tutti i valori nella collezione.

- **numpy (np):** modulo contenente le principali funzioni matematiche di appoggio, oltre che particolari strutture dati come gli array (non naturalmente presenti in python)

Segnatura	Descrizione
<code><array>.transpose()</code>	Restituisce il trasposto dell'array/matrice.
<code>arange(min, max, step=1)</code>	Restituisce un array contenente i valori che vanno da 'min' a 'max' (escluso) a scatti di 'step'.
<code>argmax(index)</code>	Restituisce l'indice dell'elemento massimo.
<code>array(object)</code>	Crea un array a partire dall'oggetto passato.
<code>linspace(min, max, num=50)</code>	Restituisce un array contenente i 'num' valori equidistanziati che vanno da 'min' a 'max' (incluso). Consigliato al posto di arange se lo step dev'essere un numero con la virgola.
<code>round(a, decimals=0)</code>	Arrotonda il valore 'a' al numero di decimali dato.

- **pandas (pd):** modulo contenente la definizione di Serie e Dataframe, utilissimi per l'organizzazione dati

Segnatura	Descrizione
<code><dataframe.at[index, column]</code>	Restituisce l'elemento nella colonna 'column' della riga indicizzata con indice 'index' (unico, no slicing).
<code><dataframe.iat[pos, column_pos]</code>	Restituisce l'elemento nella colonna in posizione 'column_pos' della riga in posizione 'pos' (same).
<code><dataframe>.apply(funzione, axis=0)</code>	Applica agli elementi del dataframe corrente la funzione data: se axis=0 alla funzione si passano le colonne, se axis=1 le righe.
<code><dataframe>.columns</code>	Estrae le colonne del dataframe.
<code><dataframe>.dropna(inplace=False)</code>	Restituisce una copia del dataframe in cui si sono eliminate le righe che hanno almeno un valore mancante (se inplace=True lavora in loco).

<code><dataframe>.groupby(column)</code>	Restituisce un dataframe che raggruppa le righe in base al valore assunto sulla colonna 'column'; su questi gruppi si possono calcolare quantità riassuntive come la media ('.mean()').
<code><dataframe>.iloc[positions]</code>	Restituisce la riga (o le righe) indicizzate tramite la loro posizione in 'positions'.
<code><dataframe>.index</code>	Estrae l'indice del dataframe.
<code><dataframe>.loc[index]</code>	Restituisce la riga (o le righe) indicizzate tramite il loro indice in 'index'.
<code><dataframe>.plot.scatter(col1, col2)</code>	Disegna il diagramma di dispersione sugli attributi 'col1', e 'col2' del dataframe (vanno dati i nomi).
<code><dataframe>.sort_index(inplace=False ascending=True)</code>	Restituisce un nuovo dataframe ordinato rispetto all'indice, oppure ordina il dataframe corrente se inplace=True.
<code><dataframe>.sort_values(by, inplace=False ascending=True)</code>	Restituisce un nuovo dataframe ordinato rispetto ai valori nella colonna 'by', oppure ordina il dataframe corrente se inplace=True.
<code><dataframe>.values</code>	Estrae i valori del dataframe.
<code><dataframe>[<esp. bool. con dataframe>]</code>	Effettua una "query" sul dataframe, restituendo un nuovo dataframe contenente solo le righe che soddisfano la condizione: nello scrivere bisogna accedere alle singole colonne del dataframe usando il suo nome come riferimento (es. <code>heroes[heroes['Height'] > 100]</code>).
<code><dataframe>[columns]</code>	Restituisce la serie corrispondente alla colonna (o alle colonne) con nome 'columns'.
<code><serie>.apply(funzione)</code>	Applica agli elementi della serie corrente la funzione passata (spesso una lambda function).
<code><serie>.corr(serie2)</code>	Restituisce il coefficiente di correlazione calcolato tra i valori della serie e quelli di 'serie2', escludendo i valori dell'indice che non compaiono in entrambi.
<code><serie>.cov(serie2)</code>	Restituisce la covarianza calcolata tra i valori della serie e quelli di 'serie2', escludendo i valori dell'indice che non compaiono in entrambi.
<code><serie>.cumsum()</code>	Genera una nuova serie in cui ad ogni indice è assegnata la somma dei valori di tutti gli indici precedenti, sé stesso incluso. Per utilizzarlo per calcolare le frequenze cumulate è necessario creare la tabella delle frequenze assolute/relative e ordinarla per indice con 'sort_index()' prima.
<code><serie>.dropna(inplace=False)</code>	Restituisce una copia della serie senza valori mancanti (se inplace=True lavora in loco).
<code><serie>.head(n)</code>	Mostra i primi 'n' valori della serie.
<code><serie>.hist(bins=10)</code>	Disegna un istogramma che rappresenta i valori nel dataframe raggruppandoli automaticamente in 'bins' gruppi distinti.

<code><serie>.iloc[p]</code>	Accede al valore in posizione 'p' della serie (anche lista di posizioni, o di valori booleani per ogni elemento considerati nell'ordine predefinito).
<code><serie>.index</code>	Estrae l'indice della serie.
<code><serie>.loc[i]</code>	Accede al valore con indice 'i' della serie (anche lista di indici, o di valori booleani per ogni elemento considerati nell'ordine predefinito).
<code><serie>.max()</code>	Restituisce il massimo valore nella serie.
<code><serie>.mean()</code>	Calcola la media sui valori della serie.
<code><serie>.min()</code>	Restituisce il minimo valore nella serie.
<code><serie>.plot() == <serie>.plot.line()</code>	Disegna con matplotlib un grafico lineare a tratti in cui gli indici sono usati come ordinate e i valori come ascisse. L'argomento opzionale <i>marker='o'</i> permette di disegnare un punto per ogni datapoint.
<code><serie>.plot.bar()</code>	Disegna con matplotlib un grafico a barre in cui sull'asse delle x si trovano gli indici (che non sono messi in scala pur se numerici) e sulle y i valori. Se chiamato su un Dataframe contenente una tabella delle frequenze relative mostra le barre affiancate di ogni valore sulle colonne per ogni valore sull'indice.
<code><serie>.plot.box(vert=True, whis=1.5)</code>	Disegna il boxplot della serie, evidenziando gli outlier con un pallino: perché i baffi ricoprano l'intero range è necessario porre 'whis=(0,100)' e per vedere il box in orizzontale serve 'vert=False'.
<code><serie>.plot.pie()</code>	Disegna un aereogramma dove gli indici sono utilizzati come label e i valori come frequenze assolute. L'argomento <i>colors</i> permette di specificare il colore per ogni label.
<code><serie>.quantile(q)</code>	Restituisce il q-esimo quantile della serie.
<code><serie>.sample(n)</code>	Estrae un campione casuale di dimensione 'n' dalla serie e lo restituisce come serie.
<code><serie>.sort_index(ascending=True, inplace=False)</code>	Restituisce una nuova serie ordinata in base all'indice della serie originale, o ordina la serie corrente se inplace=True.
<code><serie>.sort_values(ascending=True, inplace=False)</code>	Restituisce una nuova serie ordinata in base ai valori contenuti nell'originale, o ordina la serie corrente se inplace=True.
<code><serie>.std()</code>	Restituisce la deviazione standard della serie.
<code><serie>.tail(n)</code>	Mostra gli ultimi 'n' valori della serie.
<code><serie>.unique()</code>	Restituisce l'array dei valori unici nella serie.
<code><serie>.value_counts(normalize=False)</code>	Restituisce una nuova serie in cui gli indici sono i valori osservati e i valori le corrispondenti frequenze assolute (o relative se normalize=True), ordinate in senso non decrescente.

<code><serie>.values</code>	Estrae i valori della serie.
<code><serie>.var()</code>	Restituisce la varianza campionaria della serie.
<code><serie>[<esp. bool. con serie>]</code>	Effettua una “query” sulla serie, restituendo gli elementi che soddisfano la condizione booleana (<i>si utilizza il nome della serie come rappresentativo di un suo generico elemento</i>).
<code>crosstab(index, columns, colnames=None, normalize=False, margins=False)</code>	Restituisce un dataframe contenente le frequenze assolute (o relative se <code>normalize=True</code>) dei valori della serie ‘index’, a cui viene dato il nome ‘columns’ se questo contiene una stringa (<code>colnames=</code> ” serve a non visualizzare un’ulteriore etichetta per l’intera tabella). Il dataframe risulta ordinato in base all’indice. Se ‘columns’ contiene una serie restituisce invece la tabella delle frequenze (abs. o rel.) congiunte: in tal caso se <code>margins=True</code> aggiunge le righe e colonne delle frequenze marginali (in tal caso <code>normalize</code> può assumere i valori ‘all’, ‘index’ o ‘columns’).
<code>cut(serie, bins, right=True)</code>	Restituisce la serie ottenuta assegnando ogni valore in ‘serie’ all’intervallo in cui risiede tra i valori nella lista bins (aperti a dx se <code>right=False</code>).
<code>DataFrame(data, index, columns)</code>	Crea un dataframe a partire dalla matrice/dict in ‘data’ utilizzando ‘index’ come indice e dando i nomi ‘columns’ alle colonne.
<code>read_csv(path, sep=',', index_col=0, decimal='.')</code>	Restituisce il dataframe estratto dal csv/txt nella posizione ‘path’, dopo aver separato i suoi campi tramite il separatore ‘sep’ e aver utilizzato la colonna in posizione ‘index_col’ per indicizzarlo; ‘decimal’ indica il car. usato per i numeri decimali.
<code>Series(data, index=None)</code>	Crea una serie dai dati passati, indicizzando ogni valore con il relativo index (o posizione se nullo).

▪ **scipy.stats (st)**: modulo che contiene le principali distribuzioni di probabilità

Segnatura	Descrizione
<code><distribuzione_continua>.pdf(x)</code>	Restituisce il valore della funzione di densità della distribuzione calcolata in un intorno di ‘x’.
<code><distribuzione_discreta>.pmf(x)</code>	Restituisce il valore della funzione di massa di probabilità della distribuzione calcolata in ‘x’.
<code><distribuzione>.cdf(x)</code>	Restituisce il valore della funzione di ripartizione della distribuzione calcolata in ‘x’.
<code><distribuzione>.ppf(q)</code>	Restituisce il quantile q-esimo della distribuzione.
<code><distribuzione>.rvs(n)</code>	Genera un campione casuale estratto dalla distribuzione di taglia ‘n’.
<code>bernoulli(p)</code>	Crea una distribuzione bernoulliana di parametro p
<code>binom(n, p)</code>	Crea una distribuzione binomiale di parametri n e p

<code>expon(scale=1)</code>	Crea una distribuzione esponenziale con parametro $1/\text{scale}$ (per ottenere una distribuzione con λ bisogna inizializzare scale a $1/\lambda$).
<code>geom(p)</code>	Crea una distribuzione geometrica di parametro 'p'.
<code>hypergeom(N+M,N,n)</code>	Crea una distribuzione ipergeometrica di parametri $N+M=\text{num. totale di oggetti}$, $N=\text{num. di oggetti funzionanti}$ e $n=\text{num. di oggetti estratti}$.
<code>norm(loc=0, scale=1)</code>	Crea una distribuzione normale di valore atteso 'loc' e deviazione standard 'scale'.
<code>poisson(1)</code>	Crea una distribuzione di Poisson di parametro 'l'.
<code>randint(low, high)</code>	Crea una distribuzione uniforme discreta di parametro 'high – low'.
<code>uniform(loc, scale)</code>	Crea una distribuzione uniforme continua di parametri 'loc' e 'loc + scale'.

▪ **matplotlib.pyplot (plt)**: modulo standard usato per disegnare grafici a cui si appoggiano molti altri moduli, come per esempio *pandas* e *scipy* (nb: '%matplotlib inline' per mostrare i grafici nel notebook)

Segnatura	Descrizione
<code><axes>.boxplot(l, labels=None, vert=False)</code>	Disegna un boxplot per ogni serie nella lista 'l', dando a ciascuno la 'label' corrispondente. Di base questo metodo non fa preprocessing sugli outlier.
<code>bar(x, height, width=0.8)</code>	Crea un grafico a barre che traccia una barra su ogni 'x' alla relativa 'height' e con larghezza 'width'.
<code>plot(x, y)</code>	Crea il grafico di una funzione lineare a tratti associando ad ogni x la relativa y.
<code>show()</code>	Mostra tutti i grafici ancora non disegnati.
<code>step(x, y)</code>	Genera il grafico di una funzione a gradini.
<code>subplots()</code>	Restituisce una figura e un Axes che può essere usato per porre più grafici in una stessa figura.
<code>vlines(x, ymin, ymax)</code>	Crea un grafico contenente linee verticali su ogni 'x' tracciate da 'ymin' a 'ymax' (liste di valori).
<code>xlim((min, max))</code>	Impone il limite dei valori mostrati sull'asse x.
<code>xticks(ticks=None, labels=None)</code>	Determina dove tracciare i tick sull'asse delle x tramite i valori della collezione 'ticks' (vuoto se non viene passato nulla), e impone su tali tick le 'label'.
<code>ylim((min, max))</code>	Impone il limite dei valori mostrati sull'asse y.

▪ **statsmodels.api**: modulo che contiene vari strumenti di supporto per l'analisi dei dati

Segnatura	Descrizione
<code>distributions.ECDF(serie)</code>	Restituisce un oggetto di tipo funzione che calcola la funzione cumulativa empirica dell'insieme di osservazioni in 'serie'. Per calcolare la ecdf bisogna memorizzare l'output in una variabile e usarlo come una funzione su un elemento o una lista di elementi della serie originale.

<code>qqplot(data, line=False, dist=scipy.stats.distributions.norm)</code>	Disegna il diagramma QQ a partire dai quantili empirici derivati dalla serie 'data' e da quelli teorici calcolati a partire dalla distribuzione 'dist', di default una distribuzione normale ma alterabile passando altre distribuzioni da <code>scipy.stats</code> .
<code>qqplot_2samples(sample1, sample2, line=False, xlabel='', ylabel='')</code>	Disegna il diagramma QQ delle due serie passate: se <code>line='45'</code> traccia inoltre la bisettrice del primo e del terzo quadrante, mentre se <code>line='s'</code> si riporta in un ambiente standardizzato (presupponendo che i dati siano stati standardizzati).

▪ **sklearn**: modulo di machine learning contenente vari classificatori

Segnatura	Descrizione
<code><classificatore>.fit(X, Y)</code>	Genera il mapping nel classificatore naive fittando le predizioni Y sulle osservazioni X.
<code><classificatore>.predict(X)</code>	Restituisce le label predette sull'input X.
<code><LabelEncoder>.fit(serie)</code>	Genera il mapping tra etichette e valori numerici.
<code><LabelEncoder>.transform(serie)</code>	Restituisce la serie trasformata dopo aver applicato il mapping precedentemente imparato con 'fit'.
<code>naive_bayes.GaussianNB()</code>	Crea un classificatore naive di Bayes.
<code>preprocessing.LabelEncoder()</code>	Crea un nuovo LabelEncoder per trasformare etichette generiche in etichette numeriche.
<code>tree.DecisionTreeClassifier(criterion='gini')</code>	Crea l'oggetto dietro un albero di decisione, assegnando già il criterio con cui verrà trainato (il default è l'indice di gini, ma può essere cambiato all'entropia con <code>criterion='entropy'</code>).

▪ **itertools**: modulo contenente vari metodi per il calcolo combinatorio

Segnatura	Descrizione
<code>combinations(list, r)</code>	Ritorna un oggetto da tramutare in lista tramite un esplicito cast che contiene in una serie di tuple le combinazioni senza ripetizione degli elementi di 'list' in su 'r' posti.
<code>permutations(list, r)</code>	Ritorna un oggetto da tramutare in lista tramite un esplicito cast che contiene in una serie di tuple le disposizioni senza ripetizione degli elementi di 'list' in su 'r' posti (dunque anche le permutazioni).
<code>product(list, repeat=1)</code>	Ritorna un oggetto da tramutare in lista tramite un esplicito cast che contiene in una serie di tuple le disposizioni con ripetizione degli elementi di 'list' in su 'repeat' posti.