Tuple

Le tuple consentono di rappresentare sequenze **immutabili** di qualsiasi tipo di dato. Sono quindi del tutto simili alla liste, ma non possono essere modificate (come le stringhe, che però sono sequenze di soli caratteri).

```
>>> coordinatePunto = (10, -3, 4)
```

E' possibile anche non usare le parentesi. La precedente è equivalente a

```
>>> coordinatePunto = 10, -3, 4
```

Molti degli operatori e delle funzioni che abbiamo introdotto fino ad ora per le liste posso essere usati anche sulle tuple: `+`, `==`, `!=`, `in`, `not in`, tupla`[`indice`]`, tupla`[`indice1`:`indice2`]`, `len`, `min`, `max`, `sum`...

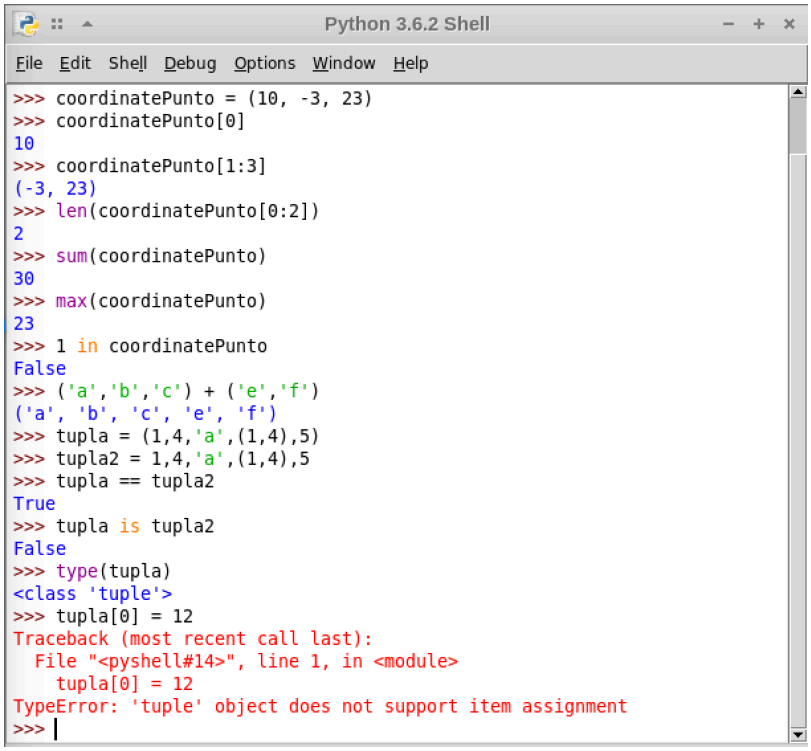
Tuple di dimensione 1

Per evitare ambiguità, una tupla di dimensione 1 deve essere scritta con una virgola dopo il primo (ed unico) elemento (senza la virgola, infatti, il comando viene considerato come una espressione fra parantesi)

```
>>> a = (1,)
>>> a
(1,)

>>> a = (1)
>>> a
1
```

Tuple: esempi di utilizzo



Assegnamento a tuple

In Python si possono assegnare valori a più variabili usando un unico enunciato

```
>>> (prezzo, quantita) = (19.95, 12)
```

Come abbiamo già detto questo si può scrivere senza parentesi

```
>>> prezzo, quantita = 19.95, 12
```

Formalmente si tratta sempre di una assegnazione fra tuple.

Da notare che questo non aggiunge potere espressivo al linguaggio. La precedente istruzione è del tutto equivalente a scrivere

```
>>> prezzo = 19.95
>>> quantita = 12
```

Si noti anche che l'assegnazione "doppia" con un unico enunciato è possibile anche usando una lista

```
>>> [prezzo, quantita] = [19.95, 12]
```

Funzioni che restituiscono più valori usando una tupla

Ovviamente una tupla può essere restituita anche da una funzione, esattamente come avviene per una lista.

```
def leggiData() :
    print("Inserisci una data")
    giorno = int(input("inserisci il giorno: "))
    mese = int(input("inserisci il mese: "))
    anno = int(input("inserisci l'anno: "))
    return (giorno, mese, anno)
```

Dalla shell dei comandi è ad esempio possibile invocare la precedente funzione come segue

```
>>> (g,m,a) = leggiData()
```

Oppure

```
>>> data = leggiData()
```

La variabile `data` si riferisce ad un contenuto immutabile, per cui non sarà possibile modificare i valori del giorno, mese ed anno.

Funzioni con un numero variabile di argomenti

In Python è possibile definire funzioni con un numero non fissato di argomenti. Supponiamo di voler definire la funzione `somma` che calcola la somma dei propri argomenti indipendentemente da quanti siano, in modo da avere, ad esempio, il seguente comportamento

```
>>> a = somma (4, 5)
>>> b = somma (5, 7, 1, 3)
>>> a
9
>>> b
16
```

La funzione deve essere dichiarata in questo modo

```
def somma (*values)
```

L'asterisco prima della variable indica che la funzione può ricevere un numero di argomenti qualsiasi. La variabile parametro `values` è in effetti una tupla che contiene tutti gli argomenti che vengono passati alla funzione

La funzione Somma Valori

```
def somma (*valori) :
    totale = 0
    for elemento in valori :
        totale = totale + elemento

    return totale
```

Si noti che invocando somma senza argomenti il risultato restituito è 0

```
>>> somma()
0
```