variabili

June 5, 2023

0.1 Variabili

Fino ad ora abbiamo esplorato Python interattivamente, e abbiamo visto come usarlo per fare calcoli o operare in maniera primitiva su dati. Osservate però che il risultato delle espressioni **svanisce** dopo essere stato mostrato. Non avrebbe più senso - memorizzarlo - utilizzarlo in altre espressioni?

In effetti quello che si può fare è associare il risultato di un'espressione ad un *nome*. In questo modo si può richiamare quel valore per una nuova epsressione. Questo è il concetto di **variabile**.

Ad esempio memorizzo il risultato di `(122-54)//7` nella variable chiamata (con molta fantasia) `risultato`. Poi posso riutilizzarla in un'altra espressione e in un'altra ancora.

```
risultato = ( 122 - 54 ) // 7
risultato * 'ciao'
```

Delle due righe precedenti la prima è un **assegnamento** ovvero una espressione è assegnata alla variabile `risultato`. La seconda riga è un'espressione come quelle che avete visto, ma nella quale invece di usare un numero, una stringa, o una funzione, utilizzo il nome della variable. Potete provare a immaginare il risultato dell'ultima espressione, e poi verificarlo in

[3]:
```
risultato = ( 122 - 54 ) // 7
```

[4]:
```
risultato * 'ciao'
```

[4]:
```
'ciaociaociaociaociaociaociaociaociao'
```

Osservate che 1. L'assegnamento non produce nessun risultato. Non è un'espressione. 2. La variable `risultato` vale `9`, e quindi il valore della seconda espressione è lo stesso che `9 * 'ciao'`.

Le variabili vengono **sostituite** all'interno di un espressione con il valore ad esse associato. Provate con espressioni differenti e più complesse. La sintassi per l'assegnamento è:

[ ]:
```
nome_variabile = espressione
```

che associa **il valore di `espressione`** alla variabile `nome_variable`. Siete liberi di scegliere il nome della variabile arbitrariamente (con poche limitazioni che vedremo).

Le variabili possono essere associate a valori di ogni tipo

[5]:
```
a = 4
b = 'ciao'
```

```
c = 4.3
```

0.1.1 Esempio: area di un cerchio

Sappiamo che l'area di un cerchio è raggio al quadrato per `3.14` (circa). Possiamo calcolare facilmente l'area di un cerchio di raggio arbitrario usando l'espressione `raggio * raggio * 3.14` qualunque sia il raggio. **Riutilizziamo** la stessa espressione per raggi diversi. La variabile **raggio** astrae il valore specifico e quindi permette di scrivere una formula che vale per qualunque raggio.

[6]:
```
raggio = 6.2
area = raggio * raggio * 3.14
area
```

[6]:
```
120.70160000000001
```

[7]:
```
raggio = 2.4
area = raggio * raggio * 3.14
area
```

[7]:
```
18.0864
```

Esercizio: scrivere delle istruzioni python analoghe, che però calcolino la circonferenza di un cerchio il cui raggio è memorizzato dalla variabile `raggio`.

0.1.2 Uso di una variabile

Una variabile, come abbiamo già detto, può essere usata (dopo essere stata definita con un assegnamento) in qualunque espressione al posto del valore ad essa associata.

[9]:
```
nome = 'Massimo'
len(nome) + 4
```

[9]:
```
11
```

[10]:
```
nome*2 + 2*'Gianni'
```

[10]:
```
'MassimoMassimoGianniGianni'
```

[11]:
```
len(nome + 'Gianni')
```

[11]:
```
13
```

[8]:
```
nome + 3
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Input In [8], in <cell line: 5>()
      3 nome*2 + 2*'Gianni'
```

```
      4 len(nome + 'Gianni')
----> 5 nome + 3

TypeError: can only concatenate str (not "int") to str
```

Fate molta attenzione: una variabile **NON** è una stringa di testo. Esiste un mondo di differenza tra `variabile` e `"variabile"`.

[12]:
```
"variabile"
```

[12]:
```
'variabile'
```

[13]:
```
variabile
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Input In [13], in <cell line: 1>()
----> 1 variabile

NameError: name 'variabile' is not defined
```

[14]:
```
variabile = 123421
```

[15]:
```
"variabile"
```

[15]:
```
'variabile'
```

[16]:
```
variabile
```

[16]:
```
123421
```

0.1.3 Evoluzione di una variable

Come avete visto, se proviamo ad usare una variabile che non è stata ancora definita, python ci da errore `NameError`. Visto che nessun valore è associato a quel nome, l'uso di quel nome causa errore.

[17]:
```
assente * 4
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Input In [17], in <cell line: 1>()
----> 1 assente * 4

NameError: name 'assente' is not defined
```

[18]:
```
assente = 1.7
assente * 4
```

[18]:
```
6.8
```

Attenzione: per usare una variabile bisogna riferirsi ad essa con la grafia corretta.

[ ]:
```
unNome = 1.7
```

[ ]:
```
unnome
```

[ ]:
```
Unnome
```

[ ]:
```
unNmoe
```

[ ]:
```
unNome
```

Una volta assegnata possiamo modificare il valore associato ad una variabile. Osservate che in python potere assegnare anche un valore di tipo diverso.

[19]:
```
nome = 'Massimo'
nome * 3
```

[19]:
```
'MassimoMassimoMassimo'
```

[20]:
```
nome = 32.5
nome * 4
```

[20]:
```
130.0
```

È **importante** capire che la modifica di una variabile ha effetto solo sulle espressioni che la coinvolgono successivamente. Le espressioni già calcolate con cambiano di valore.

[21]:
```
soggetto = 'Luisa'
saluto = "Buongiorno, " + soggetto
soggetto = 'Marco'
saluto
```

[21]:
```
'Buongiorno, Luisa'
```

0.1.4 Riassumendo

Abbiami visto che possiamo - **salvare** il risultato di espressioni nelle variabili - **riutilizzare** questi risultati in altre espressioni - **cambiare** il valore associato ad un variabile.