tipi-di-dati-e-conversioni

June 5, 2023

0.1 Tipi di dati e conversioni

Abbiamo costruito espressioni che potevano avere come valore: - una stringa di testo, e.g., `"Ciao"`; - un numero intero, e.g., `23` ; - un numero con parte decimale, e.g., `23.1534`.

In Python il valore di ogni espressione ha un *tipo* ovvero fa parte di una famiglia di valori che hanno caratteristiche in comune. Fino ad ora abbiamo visto i seguenti tipi.

- Il tipo `str` delle stringhe di testo.
- Il tipo `int` dei numeri interi (precisione infinita).
- Il tipo `float` dei numeri decimali (precisione limitata).

Il tipo di un dato indica **in quali espressioni e contesti** quel dato può essere utilizzato.

Esempi: - l'operatore `+` può essere usato mischiando `int` e `float`; - l'operatore `+` può essere usato tra `str` e `str`; - l'operatore `+` **non** può essere usato tra `str` e `int`; - l'operatore `*` può essere usato tra `str` e `int`; - l'operatore `*` **non** può essere usato tra `str` e `str`; - la funzione `len` applicata ad un parametro `str` produce un `int`.

Nel caso in cui viene effettuata un'operazione su dati del tipo sbagliato, Python segnala un `TypeError`. Provate gli esempi descritti.

[1]:
```
"ciao"*3
```

[1]:
```
'ciaociaociao'
```

[2]:
```
1 + 4 + len("abcdefg") + 4.2 + 5
```

[2]:
```
21.2
```

[5]:
```
"ciao" * "ciao"
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Input In [5], in <cell line: 1>()
----> 1 "ciao" * "ciao"

TypeError: can't multiply sequence by non-int of type 'str'
```

[3]:
```
max(23, 12.4, 1)
```

[3]: 23

[4]:
```
max(12, 'aa')
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Input In [4], in <cell line: 1>()
----> 1 max(12, 'aa')

TypeError: '>' not supported between instances of 'str' and 'int'
```

Osservate che *alcune operazioni* cambiano di comportamento a seconda dei tipi ai quali vengono applicate. Ad esempio la moltiplicazione

[7]:
```
4 * 4
```

[7]: 16

[6]:
```
'quattro' * 4
```

[6]: 'quattroquattroquattroquattro'

0.2 Conversione di tipi

Consideriamo la stringa `"12.54"`. Questa è un dato di tipo `str`, e dal punto di vista formale non è che l'elenco dei caratteri `'1'`, `'2'`, `'.'`, `'5'` e `'4'`.

Non ha nulla a che vedere con un dato di tipo `float`, eppure è possibile convertire questa stringa in un numero di tipo `float`. Python ha delle funzioni per convertire **quando possibile** un valore verso un dato tipo

- `float` trasforma qualcosa in un numero decimale;
- `int` trasforma qualcosa in un intero;
- `str` trasforma qualcosa in una stringa.

Come vedete le funzioni di conversione hanno lo stesso nome dei tipi verso cui convertono.

Esempi di conversioni verso interi

[ ]:
```
int(4.9)
```

[ ]:
```
int(1246.346)
```

[ ]:
```
int(4.0)
```

[ ]:
```
int("1000")
```

[ ]:
```
int("100" + "000")  + 23
```

[ ]:
```
int("la vispa teresa")
```

[ ]:
```
int("12.905")
```

[ ]:
```
float(412)
```

[ ]:
```
float("12.42")  - 0.21
```

[ ]:
```
float("1242")
```

[ ]:
```
float("la vispa teresa")
```

Esempi di conversione verso stringa

[ ]:
```
str(132892)
```

[ ]:
```
str(13) + str(4)
```

[ ]:
```
str(13 + 4)
```

[ ]:
```
str(-14.5)
```

0.2.1 Riassumendo

Abbiamo visto - che ogni dato ha un **tipo** - gli operatori e le funzioni si **comportano** diversamete a seconda dei tipi - l'uso di tipi incompatibili **causa errore** - è possibile **convertire** un dato in un dato di tipo diverso