espressioni-sempre-piu-complesse

June 5, 2023

0.1 Espressioni sempre più complesse

Ricapitoliamo i tipi visti fino ad ora

- stringhe `str` (e.g. `"ciao"`, `"miao"`, `"Elefante rosa!"`)
- interi `int` (e.g `22`, `42`, `7`, `100438`, `-42423`)
- numeri approssimati `float` (e.g. `13.4`, `-0.342`, `.45`)
- booleani `bool` (e.g. `True` e `False`)

Questi tipi possono essere usati per costruire espressioni tramite

- parentesi tonde `(`, `)` che possono essere annidate senza limite;
- chiamate a funzioni o metodi , e.g., `len("coniglio")`, `"max".upper()`;
- operatori aritmetici `+`, `-`, `*`, `/`, `//`, `%`, `**`;
- operatori di confronto `==`, `!=`, `<`, `>`, `<=`, `>=`;
- operatori logici `not`, `and`, `or`.

Esercizio: riuscite a calcolare il valore della sequente espressione senza eseguirla nell'interprete Python?

[1]:
```
not -5//2**4 < -1 and 3 ** 2 ** (5 + - 3) >= 2*len("casa")
```

[1]:
```
True
```

0.1.1 Precedenze degli operatori

Proabilmente l'ostacolo più grande nell'esercizio precedente è sapere quali operatori applicare prima o dopo. Le precedenze in Python sono più o meno le stesse di altri linguaggi di programmazione, e generalizzando le precedenze usate in algebra. Elenchiamo gli operatori in ordine, da quelli con maggiore priorità a quelli con minore priorità.

1. Parentesi `(`, `)`;
2. Funzioni o metodi;
3. Operatori aritmentici in quest'ordine:
    1. esponenziazione `**`, valutato da destra a sinistra;
    2. segni `+` e `-` dei numeri, per esempio `-2` e `+2.4`;
    3. moltiplicazioni e divisioni `*`, `/`, `//`, `%` da sinistra a destra;
    4. somme e sottrazioni `+`, `-`, da sinistra a destra.
4. Confronto `<`, `>`, `>=`, `<=`, `==`, `!=`
5. Logici:
    1. `not`

2. `and` da sinistra a destra;
3. `or` da sinistra a destra.

Regole più precise si trovano nella documentazione Python.