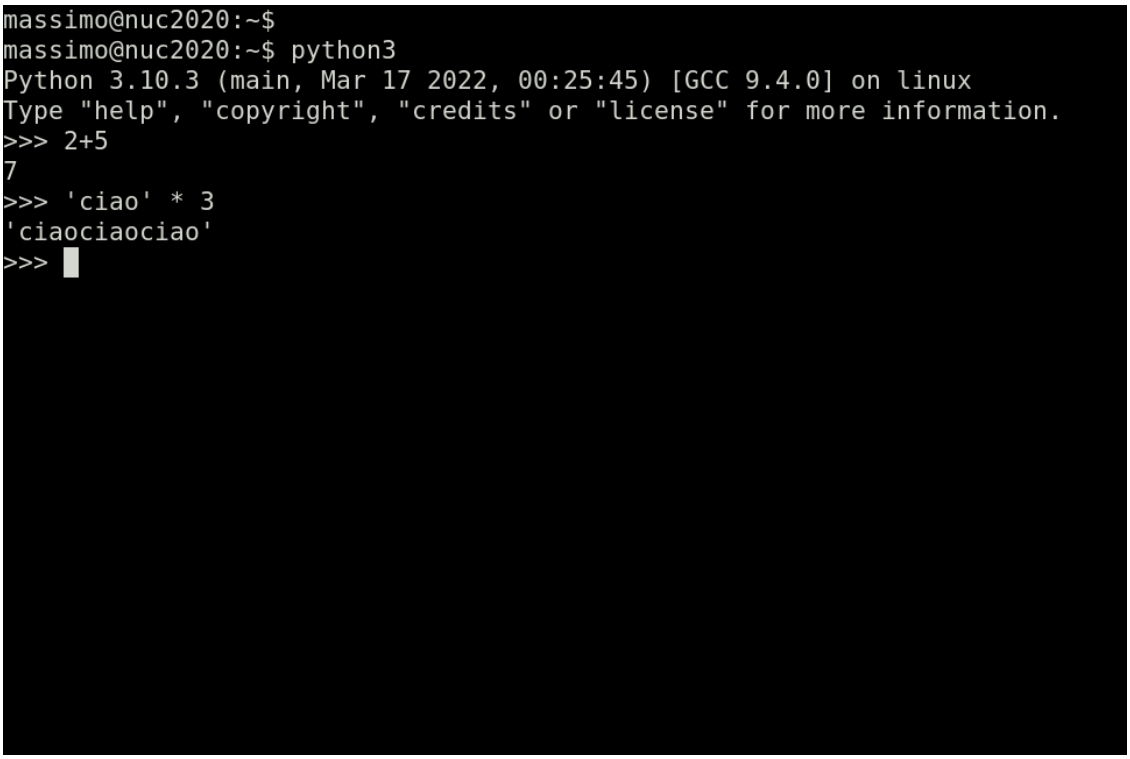
sessione-interattiva

June 5, 2023

0.1 Sperimentare con Python interattivo

Per fare i primi passi con Python non è necessario scrivere un programma, possiamo farlo direttamente nella sessione interattiva.



Questa sessione è essenzialmente la stessa che ottenete lanciando il comando `python3`, seguito da invio, sul terminale. In quel caso i caratteri `>>>` indicano che python è pronto per fare tutto quello che gli dite.

Per cominciare scrivere `2+5` e premete invio

[1]:
```
2+5
```

[1]:
```
7
```

La sessione interattiva permette di scrivere espressioni e comandi python. A ogni espressione inserita python risponde con il suo risultato.

Quali espressioni e comandi? Per esempio - numeri interi `48239`, `120000000`, - numeri decimali `-34.632` - espressioni costruite utilizzando operatori `+`,`-`,`*`,`/` e parentesi tonde.

Sperimentate!

[ ]:

Provate a scrivere altre espressioni - che succede se annidate più parentesi tonde e.g. `2*(2.4 - (1.0+2)*5)`;? - che differenza c'è tra `1.5 + 4 * 2`, `1.5 + (4 * 2)`, e `(1.5 + 4) * 2`?

[2]:
```
-4 + (2.5 - 4 / 2)
```

[2]:
```
-3.5
```

Potete usare Python come **calcolatrice!**

0.1.1 Suggerimento

Nella sessione interattiva, usando le freccie Su e Giù potete richiamare i comandi che avete già eseguito, e li potete ripetere o modificare.

0.1.2 Stringhe di testo

I linguaggi di programmazione permettono di elaborare numeri ma anche di manipolare testi. Una *stringa* è un dato che rappresenta una sequenza di caratteri testuali. L'espressione `"Gatto"` in python è interpretata come un dato il cui valore è una sequenza di cinque caratteri di cui il primo è la g maiuscola, il secondo la a minuscola, ecc…

[3]:
```
'Gatto'
```

[3]:
```
'Gatto'
```

[4]:
```
"Andrà tutto bene"
```

[4]:
```
'Andrà tutto bene'
```

[5]:
```
"I cavalieri" + " della tavola rotonda"
```

[5]:
```
'I cavalieri della tavola rotonda'
```

[6]:
```
5 * "Ohm"
```

[6]:
```
'OhmOhmOhmOhmOhm'
```

[7]:
```
"Tora!" * 3
```

[7]:
```
'Tora!Tora!Tora!'
```

Come potete vedere le stringhe non solo rappresentato testi, ma possono essere manipolate, ad esempio con l'operatore `+` che concatena due stringhe, o l'operatore `*` che ripete una stringa.

Lo stesso operatore ha significati diversi a seconda dei dati su cui opera - l'operatore + tra due numeri, li somma - l'operatore + tra due stringhe, produce la stringa concatenata - l'operatore + tra un numero e una stringa produce errore

[8]:
```
5 + "lune"
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Input In [8], in <cell line: 1>()
----> 1 5 + "lune"

TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

0.1.3 Piccola complicazione: apici singoli e doppi

Le stringhe devono essere specificate inziando con un apice (singolo o doppio) e si concludono quando si incontra lo stesso tipo di apice.

[ ]:
```
"stringa con apici doppi"
```

[ ]:
```
'stringa con apici singoli'
```

[ ]:
```
uno dei due apici è obbligatorio
```

[ ]:
```
"doppio"  + 'singolo'
```

[ ]:
```
"questa stringa contiene un apice ' che è singolo"
```

[ ]:
```
'questa stringa contiene un apice " che è singolo'
```

Domanda perché l'espressione seguente causa un errore?

[ ]:
```
'scova l'errore'
```

Se voglio mettere un apice singolo un una stringa delimitata da apici singoli lo posso inserire usando \', e posso fare lo stesso con un apice doppio usando \".

[ ]:
```
"Apice doppio \" e apice singolo \', insieme."
```

[ ]:
```
'Apice doppio \" e apice singolo \', insieme.'
```

0.1.4 Riassumendo

Abbiamo visto - come usare Python **interattivamente** - dati numerici e stringhe - operatori per costruire espressioni complesse - apici singoli e doppi - segnalazioni di **errori** nella sessione interattiva.

[ ]:
```

```