

1. Introduzione a Python

Corso di Python per il Calcolo Scientifico

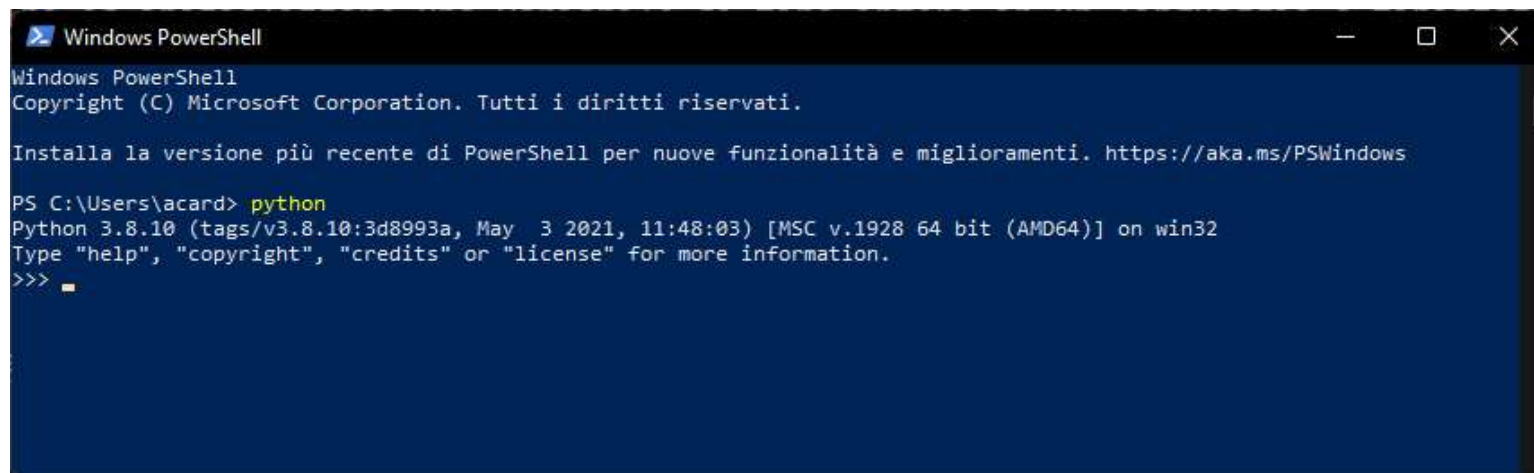
Angelo Cardellicchio
angelo.cardellicchio@stiima.cnr.it

Outline

- L'interprete Python
- Python e tipizzazione
- Duck Typing
- Operazioni aritmetiche
- Stringhe
- Liste

L'interprete Python

- A differenza del C, il Python è un linguaggio **interpretato**.
- Una volta installato l'interprete, potremo richiamarlo direttamente da riga di comando ed interagirvi.
 - *Ciò non implica che non sia possibile scrivere dei programmi 'classici'!*



```
Windows PowerShell
Copyright (C) Microsoft Corporation. Tutti i diritti riservati.

Installa la versione più recente di PowerShell per nuove funzionalità e miglioramenti. https://aka.ms/PSWindows

PS C:\Users\acard> python
Python 3.8.10 (tags/v3.8.10:3d8993a, May  3 2021, 11:48:03) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> █
```

Python e tipizzazione

- Python è un linguaggio a **tipizzazione dinamica**.
- Ciò implica che il tipo di una variabile è **direttamente inferito** dall'interprete, per cui non è necessario specificarlo al momento della sua dichiarazione.
- Un'istruzione di questo tipo è quindi perfettamente valida:

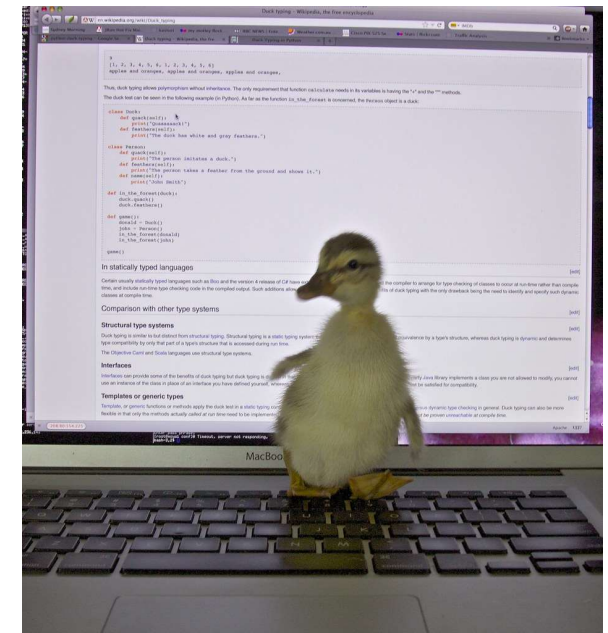
```
var = 0
```

- Ciò rende anche implicito il casting. Infatti, possiamo sommare una variabile intera ad una di tipo float senza fare alcuna operazione di casting (esplicita).

```
var + 1.1    # Il risultato sarà 2.1
```

Duck Typing

- *If it walks like a duck, and it quacks like a duck, then it must be a duck.*
- L'interprete 'stabilisce' il tipo in base al suo 'comportamento'.
- Se la variabile si 'comporta' come un intero, deve essere *un intero*.



Operazioni aritmetiche (1)

- Le divisioni restituiscono sempre un numero in virgola mobile.

```
>>> 16 / 3
5.333333333333333
>>> 2 / 2
2.0
```

- Vi sono operatori per il **quoziente** ed il **resto**.

```
>>> 16 // 3
5
>>> 16 % 3
1
```

Operazioni aritmetiche (2)

- L'elevazione a potenza prevede l'uso dell'operatore `**`.

```
>>> 3 ** 2
```

```
9
```

```
>>> 2 ** 8
```

```
256
```

- Le restanti operazioni sono simili a quelle fatte in C.

```
>>> 2 + 2
```

```
4
```

```
>>> 3 * 5
```

```
15
```

Stringhe (1)

- Le stringhe possono essere indifferentemente racchiuse tra virgolette singole e doppie.

```
>>> "una stringa"  
'una stringa'  
>>> 'un\'altra stringa'  
"un'altra stringa"
```

- Non vanno mischiate!

```
>>> 'un'altra stringa'  
File "<stdin>", line 1  
'un'altra stringa  
SyntaxError: invalid syntax
```


Stringhe (2)

- Le stringhe possono essere su più righe.

```
>>> print("""Questo è un esempio  
di  
riga multipla  
""")
```

- Le stringhe possono essere concatenate.

```
>>> stringa_a = "Prima stringa"  
>>> stringa_b = "Seconda stringa"  
>>> print(stringa_a + " - " + stringa_b)  
Prima stringa - Seconda stringa
```

Stringhe (3)

- Le stringhe sono considerate degli array, e quindi indicizzabili.

```
>>> stringa = 'Python'
>>> stringa[0]
'P'
```

- Python permette di accedere agli elementi negativi.

```
>>> stringa[-1]
'n'
```

- Python permette anche lo **slicing**.

```
>>> stringa[0:2]
'Py'
>>> stringa[2:5]
'tho'
```

Stringhe (4)

- La lunghezza di una stringa è data dalla funzione **len()**.

```
>>> len(stringa)
6
```

- Una stringa è **immutabile**.

```
>>> stringa[0] = 'C' # Errore!
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'str' object does not support
item assignment
```

Liste (1)

- Una lista è 'simile' agli array di C.

```
>>> lista = [1, 2, 3, 4, 5]  
[1, 2, 3, 4, 5]
```

- Le operazioni viste sulle stringhe valgono anche sulle liste.

```
>>> lista[0]  
1  
>>> lista[2:]  
[3, 4, 5]  
>>> lista_due = [6,7]  
>>> lista + lista_due  
[1, 2, 3, 4, 5, 6, 7]  
>>> lista + [6]  
[1, 2, 3, 4, 5, 6]
```

Liste (2)

- Una lista è mutabile.

```
>>> lista[0] = 99  
>>> lista  
[99, 2, 3, 4, 5]
```

- È possibile aggiungere elementi delle liste.

```
>>> lista.append([1,2,3])  
>>> lista  
[99, 2, 3, 4, [1, 2, 3]]
```

Domande?

42