

Il linguaggio Python

Liceo G.B. Brocchi

Classe 3AQSA – Compresenza Informatica - Arte
Bassano del Grappa, Dicembre 2022

Il vostro futuro linguaggio preferito

Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms.

Il vostro futuro linguaggio preferito

- Creato da **Guido van Rossum**
https://en.wikipedia.org/wiki/Guido_van_Rossum a fine anni '80
- La sintassi è molto semplice: ricorda uno *pseudocodice con parole chiave in inglese*
- È un linguaggio **ad alto livello**: anche il C e il C++ sono ad alto livello, ma sono molto più vicini all'hardware rispetto a Python
- Python può essere utilizzato su qualsiasi sistema
- Le librerie disponibili per Python sono moltissime e facili da usare: potete usare Python per fare machine learning, sviluppo web, crittografare informazioni, fare calcolo scientifico etc..

I linguaggi compilati

Il C e il C++ sono linguaggi compilati. Il codice sorgente viene convertito in codice macchina da un compilatore. Il risultato della compilazione (e del linkaggio), ossia un file eseguibile, viene caricato in memoria RAM ed eseguito dalla CPU, ossia dall'hardware del computer

codice sorgente C (testo ASCII)

compilatore/linker

file binario eseguibile

I linguaggi interpretati

Python, JavaScript, PHP sono linguaggi interpretati.
Nel caso di Python, l'interprete (Python) esegue direttamente il codice sorgente.
In realtà esiste un processo di traduzione in codice macchina interno a Python, ma al programmatore non interessa.
Il programmatore deve solo scrivere il codice e chiedere a Python di eseguirlo.
È abbastanza simile a quello che succede quando la shell interpreta i vostri comandi

codice sorgente Python (testo ASCII)



l'interprete esegue il codice

Interprete interattivo vs file sorgente

- Potete utilizzare Python dall'interprete interattivo
 - lanciate semplicemente il comando **python**, o **python3** da una shell Linux, o dal prompt di Windows
 - si aprirà una shell in grado eseguire istruzioni Python
 - i caratteri **>>>** costituiscono il **Python interpreter prompt**
 - l'istruzione **quit()** chiude l'interprete interattivo

```
cyofanni@LAPTOP-I0S1RKRC: /mnt/c/WINDOWS/system32
cyofanni@LAPTOP-I0S1RKRC:/mnt/c/WINDOWS/system32$ python3
Python 3.10.6 (main, Nov  2 2022, 18:53:38) [GCC 11.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> print("hello world")
hello world
>>> _
```

Interprete interattivo vs file sorgente

- Chiaramente, non ha molto senso riscrivere programmi interi ogni volta nell'interprete interattivo
- Per scrivere un programma una volta sola ed eseguirlo tutte le volte che si vuole bisogna salvarlo in un file

```
cyofanni@LAPTOP-I0S1RKRC: /mnt/c/WINDOWS/system32
cyofanni@LAPTOP-I0S1RKRC:/mnt/c/WINDOWS/system32$ python3
Python 3.10.6 (main, Nov  2 2022, 18:53:38) [GCC 11.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> print("hello world")
hello world
>>> _
```

Interprete interattivo vs file sorgente

- Chiaramente, non ha molto senso riscrivere programmi interi ogni volta nell'interprete interattivo
- Per scrivere un programma una volta sola ed eseguirlo tutte le volte che si vuole bisogna salvarlo in un file
- Scrivere il codice in un file con estensione **.py**
- Per eseguire lo il file (script): **python filename.py**

Variabili, stringhe, format()

#no need to write a main function

year = 1756 #no need to write the type

name = 'Wolfgang Amadeus Mozart' #no need to write the type, notice the single quote

work = 'The Magic Flute' #notice the single quote

print(' {0} was born in {1}'.**format**(name, year))

print('his most famous opera is {0}'.**format**(work))

Operatori matematici

- `2 * 3` restituisce 6 (ovviamente)
- `'hello' * 4` restituisce `'hellohellohellohello'` (comodo vero?)
- `2 ** 5` restituisce $32 = 2^5$
- `15 // 2` divisione intera

Operatori bitwise – left shift

- Per scrivere numeri nel sistema binario utilizziamo la sintassi **0b**

0b1111

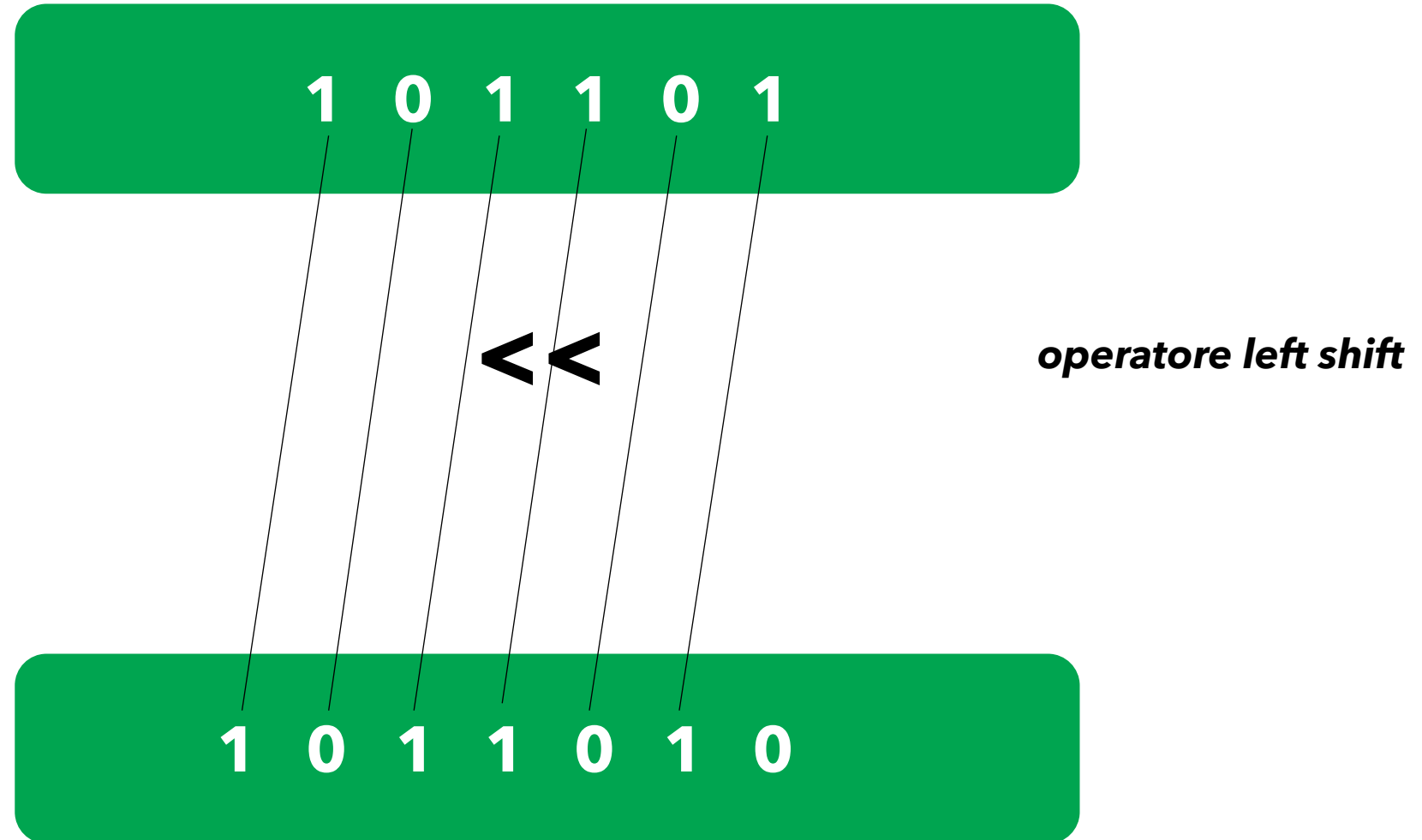
: 15

- Per convertire un numero decimale in binario, utilizziamo la funzione **bin()**

bin(255)

0b11111111

Operatori bitwise – left shift



Operatori bitwise – left shift

`0b111 << 1`

14



Python converte il risultato automaticamente in decimale. Mai noi vorremmo vederlo in binario per capire meglio cosa ha fatto l'operatore `<<` (left shift)

```
bin(0b111 << 1)
'0b1110'
```

Operatori bitwise – left shift

```
bin(0b10 << 1)  
'0b100'
```

```
bin(0b10 << 2)  
'0b1000'
```

```
bin(0b10 << 3)  
'0b10000'
```

```
2 << 1  
4
```

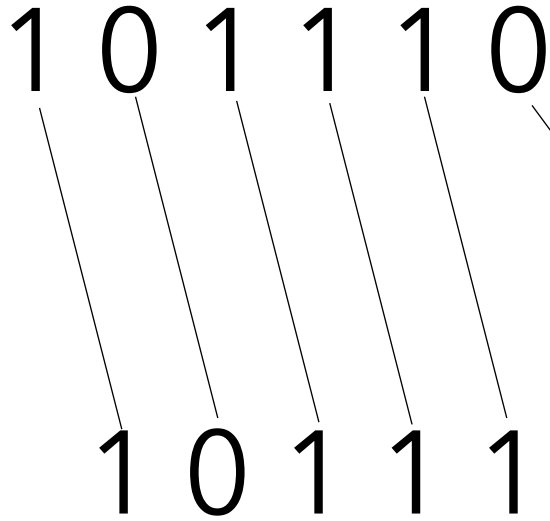
```
2 << 2  
8
```

```
2 << 3  
16
```

Un left shift di un numero b di n posizioni equivale a moltiplicare b per 2^n

Operatori bitwise – right shift

1 0 1 1 1 0
1 0 1 1 1



la cifra meno significativa viene tagliata.
Se il left shift era una moltiplicazione per 2^n , il right shift è ... provate!

Operatori bitwise booleani

- Per casa, provare questi altri operatori bitwise con l'interprete interattivo di Python:

& (AND)

| (OR)

^ (XOR)

Potete utilizzarli su numeri in decimale, ma per capirne meglio il funzionamento vi consiglio di utilizzare costanti binarie e convertire tutto in binario.

Un piccolo script

```
import math
```

```
radius = 5
```

```
sphere_area = 4 * math.pi * radius**2
```

```
sphere_volume = (4/3) * math.pi * radius**3
```

```
print('Area of sphere of radius: {0} cm, is: {1} cm^2'.format(radius, sphere_area))
```

```
print('Volume of sphere of radius: {0} cm, is: {1} cm^3'.format(radius,  
sphere_volume))
```

Usage: python3 sphere.py

Un piccolo script

```
#!/usr/bin/python3
```

```
import math
```

```
radius = 5
```

```
sphere_area = 4 * math.pi * radius**2
```

```
sphere_volume = (4/3) * math.pi * radius**3
```

```
print('Area of sphere of radius: {0} cm, is: {1} cm^2'.format(radius, sphere_area))
```

```
print('Volume of sphere of radius: {0} cm, is: {1} cm^3'.format(radius,  
sphere_volume))
```

si specifica il percorso in cui si trova l'interprete Python installato sulla vostra macchina

Bisogna utilizzare lo **shebang #!**

Prima bisogna capire dove è posizionato l'eseguibile dell'interprete Python: lanciate **which python3** dalla shell bash

Make the file executable: `chmod 755 sphere.py`

Now you can run it with: `./sphere.py`

Le strutture di controllo

```
number_str = input('enter a number: ')\nnumber_int = int(number_str)
```

l'input da tastiera (standard input) è di tipo stringa (str). Se vogliamo trattarlo come numero, dobbiamo effettuare una conversione di tipo

```
if number_int % 2 == 0:\n    print('you entered an even number')\nelse:\n    print('you entered an odd number')
```

conversione da stringa a intero (da str a int)

Le strutture di controllo

```
while True:  
    print('python is great')
```

```
i = 1  
while i <= 10:  
    i += 1  
    print(i)
```

il costrutto while è molto simile a quello
dei linguaggi C-like

Le strutture di controllo

```
for i in [2, 5, 3, 1, -12, 3, 9]:  
    print(i, '\t', end='')  
print()
```

Cos'è questo oggetto misterioso tra parentesi quadre?

il costrutto for è un po' particolare

Output: 2 5 3 1 -12 3 9

Le strutture di controllo

```
for item in ['3AQSA', '2AQSA', '3BSA', '4BSA', '1ASA', '2ASA', '3ASA']:  
    print(item, '\t', end='')  
print()
```

Output: 3AQSA 2AQSA 3BSA 4BSA 1ASA 2ASA 3ASA

Sembra proprio che il for non lavori sugli indici, ma direttamente sugli elementi dell'oggetto specificato dopo la keyword 'in'

Le liste

```
l1 = [1, 2, 3, 6, 1, 2]
```

```
l2 = [[2, 6, 3], [1, 6, 4]]
```

```
l3 = ['alan turing', 'john von neumann', 'gottfried leibniz']
```

```
l4 = ['banana', 1, 5, 3.4, l1, l2, 'strawberry']
```

le liste di python sono molto più comode e flessibili degli array dei linguaggi a livello più basso.

come potete notare, la lista l4 contiene stringhe, interi, double e liste.

Quindi le liste possono contenere elementi di tipo diverso