Introduzione a Python

Advanced Computer Programming

Prof. Luigi De Simone

Sommario

- Introduzione a Python
- Tipi scalari e non scalari, Stringhe
- Flusso di controllo: If then else, While, For
- Funzioni
- Tuple, List, Dizionari

Riferimenti

- Tony Gaddis. **Introduzione a Python**. 5° ed. – Pearson, 2021
- Paul J. Deitel, Harvey M. Deitel. **Introduzione a Python. Per l'informatica e la data science.** Pearson, 2021
- **Python: How to Think Like a Computer Scientist interactive edition** https://runestone.academy/runestone/books/published/thinkcspy/index.html
- Allen Downey. **Think Python** - https://greenteapress.com/thinkpython2/thinkpython2.pdf
- http://docs.python.org/library/ (Informazioni sulla libreria standard di Python)
- http://docs.python.org/tutorial/modules.html (Tutorial sui moduli di Python)

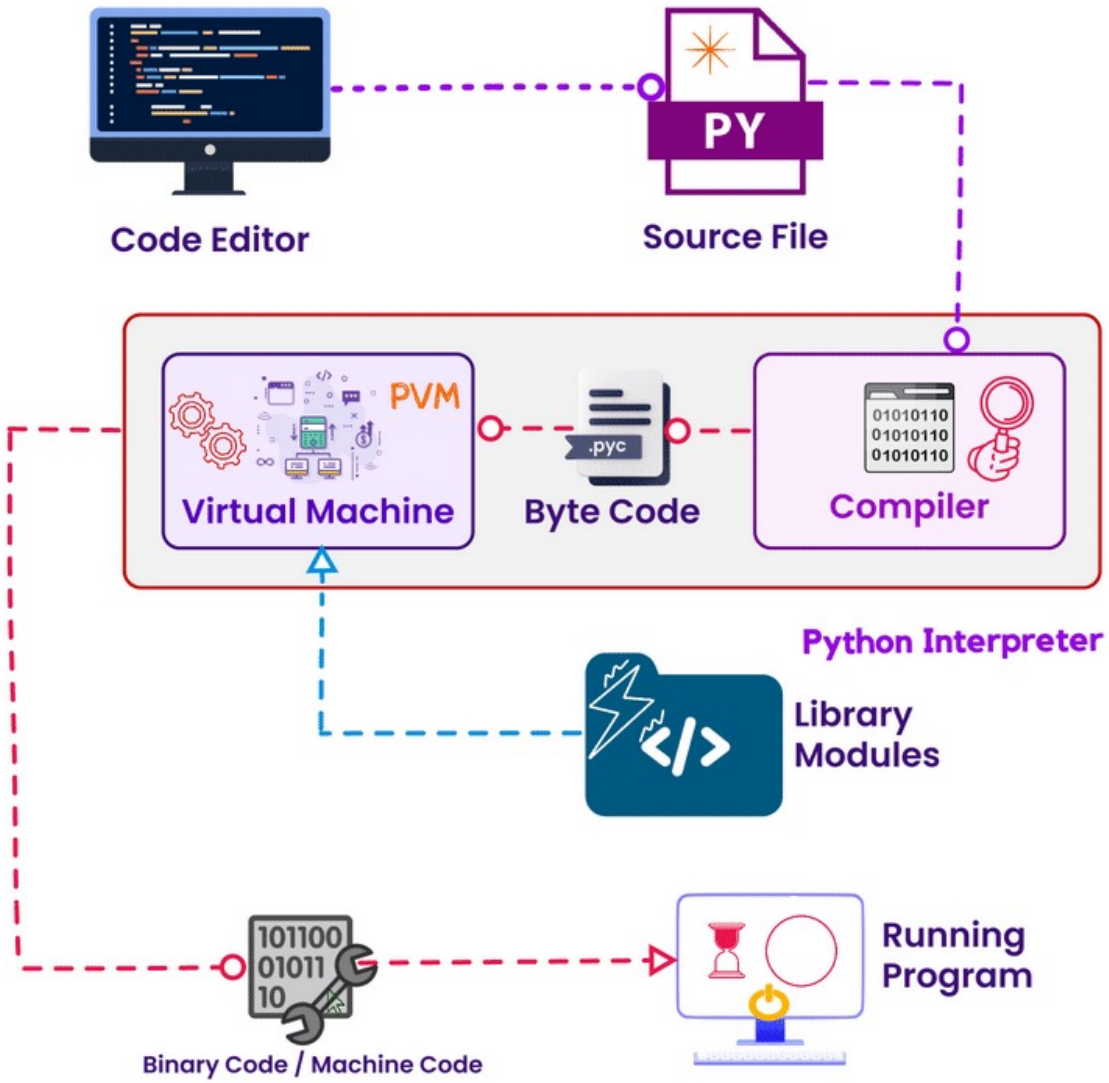
Compilatori vs Interpreti

- Programmi scritti in linguaggio di alto livello devono essere tradotti in linguaggio macchina per essere eseguiti

- **Compilatori**: traducono programmi scritti in linguaggi di alto livello in un programma separato, scritto in linguaggio macchina
  - Un programma scritto in linguaggio macchina può essere eseguito in qualunque altro momento

- **Interpreti**: traducono ed eseguono le istruzioni di un programma scritto in linguaggio di alto livello
  - **Utilizzato dal linguaggio Python**
  - Interpreta una istruzione alla volta
  - Non c'è un programma separato scritto in linguaggio macchina

Come funziona Python



Interprete Python

- L'implementazione di riferimento dell'**interprete Python** è **CPython**
  - CPython è scritto sia in C che in Python stesso

- Non converte il codice sorgente in istruzioni in linguaggio macchina ma *trasforma l'intero codice in qualcosa chiamato* ***byte code***
  - All'interno dell'interprete Python **avviene comunque una compilazione**, ma tale compilazione non converte l'intero codice in linguaggio macchina o assembly come avviene in linguaggi compilati come C/C++.
  - ***NOTA BENE: byte code*** e **istruzioni assembly** non sono la stessa cosa
    - Il **byte code** viene generato all'interno di una *virtual machine* e per una *virtuale machine* (*software di sistema intermedio*)
    - Il **linguaggio assembly** viene creato per una specifica CPU

Interprete Python

- Il **compilatore Python**
  - Legge le istruzioni del programma scritto in Python
  - Verifica se le istruzioni sono formattate correttamente, cioè verifica la struttura sintattica di ogni riga del programma
    - Se ci sono errori, la **traduzione viene immediatamente interrotta** e viene visualizzato un messaggio di errore
    - Se non ci sono errori, il compilatore traduce le istruzioni di alto livello nel linguaggio intermedio equivalente che è il ***byte code***.
- Il *byte code* viene fornito al vero è proprio interprete Python, la cosiddetta **Python Virtual Machine (PVM)**
  - La **PVM** converte il *byte code* Python in istruzioni a livello macchina o in codice binario equivalente
  - Se si verifica un errore in questa fase di interpretazione, la conversione si arresta mostrando un messaggio di errore.

Interprete Python

- **Python deve essere installato e configurato prima dell'uso**
  - Installare l'interprete Python (https://www.python.org/downloads/)

- L'interprete Python può essere utilizzato in due modalità:
  - ***Interactive mode***: scrivere le istruzioni Python ed eseguirle una alla volta
  - ***Script mode***: scrivere tutte le istruzioni e salvarle in uno script Python da eseguire in un secondo momento

Interactive mode

- Quando eseguiamo Python in *interactive mode*, utilizziamo un **prompt dei comandi**
  - L'interprete rimane in attesa di istruzioni Python da scrivere
  - Il prompt dei comandi riappare dopo l'esecuzione dell'istruzione precedente
  - Se l'istruzione è non corretta saranno mostrati eventuali messaggi di errore
- Questa modalità è un buon modo per iniziare a capire Python

Script mode

- Dobbiamo utilizzare la ***script mode*** per eseguire un intero programma Python
  - Salvare l'insieme delle istruzioni Python in un **file con estensione .py** (raccomandato)
  - **Eseguire** da terminale il file Python **invocando l'interprete**
    - `python filename.py`
  - L'interprete Python valuterà passo passo tutte le istruzioni scritte nello script
- In questa modalità vengono forniti i classici meccanismi per ottenere **argomenti da linea di comando** e/o **redirezione di I/O**
- Python possiede anche un meccanismo per consentire ai programmi Python di **agire sia da script** che da **modulo** da importare ed essere usato da altri programmi Python

Byte code compilato: **i file .pyc**

- I **file .pyc** sono file **byte code compilati** che vengono generati dall'interprete Python quando uno **script Python viene importato o eseguito**
- I file .pyc possono essere **eseguiti direttamente dall'interprete**, senza la necessità di ricompilare il codice sorgente a ogni esecuzione dello script
  - Tempi di esecuzione degli script più rapidi
- Per ottenere direttamente il file .pyc a partire da un file .py eseguire
  - `python -m compileall filename.py`

Programmi Python

- Un **programma Python** è una sequenza di definizioni e comandi
  - Le definizioni sono **valutate**
  - I comandi sono **eseguiti** dall'interprete Python in una shell (terminal)
- **I comandi** (statements) istruiscono l'interprete a fare qualcosa
- Tali comandi possono essere digitati direttamente all'interno di una **shell** oppure immagazzinati in un **file** che viene letto dalla shell per essere valutato

Python IDEs

- **VSCodium** (con estensioni Jupyter, Python)
  https://vscodium.com/
- **Anaconda Navigator**
  https://www.anaconda.com/products/individual
  - Notebook Jupiter
- **PyCharm**
  https://www.jetbrains.com/pycharm/download/
- **Terminale…**