Differenza tra Prompt e Programma

Sebbene entrambi rappresentino istruzioni fornite a un sistema di calcolo per fargli compiere un'azione, **un prompt** e **un programma** appartengono a due paradigmi computazionali profondamente differenti.

1. Definizioni di base

- **Programma:** Una sequenza formale, non ambigua e deterministica di istruzioni scritte in un linguaggio di programmazione (es. Python, C++, Java, Rust). Viene tradotto da un compilatore o interprete in codice macchina per essere eseguito direttamente dall'hardware (CPU/GPU).
- **Prompt:** Un input espressivo (solitamente in linguaggio naturale o semi-strutturato come testo, immagini o audio) fornito a un modello di intelligenza artificiale generativa (come un Large Language Model) per orientare e condizionare la generazione dell'output.

2. Tabella comparativa

Dimensione	**Programma Informatico**	**Prompt per AI**
Linguaggio utilizzato	Formale e rigoroso (sintassi rigida, tipizzazione, costrutti logici).	Naturale (italiano, inglese, ecc.) o debolmente strutturato (Markdown, JSON).
Natura del risultato	**Deterministico**: a parità di input e stato, l'output è identico al 100%.	**Probabilistico/Stocastico**: l'output si basa su stime statistiche dei token successivi.
Tolleranza all'errore	Molto bassa: un errore di sintassi o una virgola mancante causano crash o anomalie logiche.	Molto alta: tollera refusi, ambiguità lessicali e cambi di tono, interpretando il contesto.
Controllo del processo	Definisce **il "come"** passo dopo passo (algoritmo, manipolazione di memoria, cicli).	Definisce **il "cosa"** (intento, contesto, vincoli desiderati), lasciando l'inferenza al modello.
Interprete / Esecutore	Compilatore, interprete e processore deterministico.	Rete neurale profonda e pesi statistici pre-addestrati.

Debug e verifica	Tracciamento deterministico dello stack trace, test unitari formali, step-by-step debugger.	Valutazione qualitativa, raffinamento del contesto, tecniche di prompt engineering, benchmark statistici.

3. Le differenze concettuali chiave

A. Come vs. Cosa (Imperativo vs. Descrittivo)

- In un **programma**, lo sviluppatore deve specificare l'esatta logica di esecuzione:
```
# Esempio Programma: dice esattamente COME sommare e filtrare
def filtra_pari(numeri):
    risultato = []
    for n in numeri:
        if n % 2 == 0:
            risultato.append(n)
    return risultato
```

- In un **prompt**, l'utente descrive l'obiettivo o fornisce degli esempi, confidando nella capacità di ragionamento induttivo del modello:*"Dato questo elenco di numeri [1, 4, 7, 10, 15], estrai solo i numeri pari e ordinali in senso decrescente."*

B. Rigidità logica vs. Flessibilità contestuale

- I **programmi** sono ideali per calcoli matematici esatti, gestione affidabile di database, crittografia e transazioni finanziarie, dove la tolleranza al margine d'errore è zero.
- I **prompt** eccellono in compiti aperti ad alto livello di astrazione semantica: sintesi di documenti, riformulazione di testi, ideazione, traduzione stilistica ed estrazione di senso da dati disordinati.

4. Dove si incontrano: la convergenza moderna

Nel paradigma dell'intelligenza artificiale moderna, i due mondi si fondono:

1. **Il codice contiene prompt:** Gli sviluppatori integrano chiamate API a modelli di linguaggio all'interno di normali programmi (ad esempio per analizzare una recensione utente e restituire un punteggio).
2. **I prompt generano codice:** I programmatori usano i prompt per chiedere all'AI di scrivere intere funzioni o correggere bug in codice C++ o Python.
3. **Prompt come "programmazione ad alto livello":** Concetti come il *Few-Shot Prompting*, il *Chain-of-Thought* e la creazione di agenti autonomi dotati di strumenti (Function Calling) rendono il prompt engineering una vera e propria forma di architettura logica.