Introduzione IDE e Python

Python

Un IDE (Integrated Development Environment) è uno strumento che racchiude in un unico ambiente tutte le risorse necessarie per sviluppare software: dall'editor di codice alla gestione di compilazione, debug e test.

Utilizzare un IDE consente di velocizzare il flusso di lavoro, poiché si hanno a disposizione funzionalità dedicate come il completamento automatico, il refactoring e il controllo di versione.

In questo modo, chi sviluppa può concentrarsi maggiormente sulla logica del programma anziché su task ripetitivi e di contorno.

Lista dei 3 IDE più usati

- **Visual Studio Code (VSC)**

Un editor di testo avanzato ma estremamente personalizzabile, che diventa un IDE completo grazie all'installazione di estensioni. Offre integrazione con Git, debugging integrato e un marketplace ricco di plugin per tutti i principali linguaggi di programmazione, compreso Python.

- **PyCharm**

Un IDE specificamente progettato per Python, sviluppato da JetBrains. Integra strumenti di refactoring, debugging, test e gestione dei pacchetti Python. La versione Community è gratuita e copre le esigenze principali, mentre la versione Professional offre funzionalità aggiuntive, utili ad esempio per lo sviluppo web (Django, Flask).

- **Jupyter Notebook**

Un ambiente interattivo che permette di scrivere ed eseguire codice in "notebook", suddivisi in celle. È molto apprezzato per l'analisi dei dati e il machine learning, grazie alla possibilità di integrare codice, testo descrittivo (Markdown) e grafici nello stesso documento.

Visual Studio Code (spesso abbreviato in VS Code o VSC) è un editor di testo avanzato, molto diffuso grazie alla sua leggerezza e personalizzabilità.

Non è un IDE “classico” come altri ambienti (ad esempio Visual Studio o IntelliJ), ma può diventarlo grazie all’installazione di estensioni create dalla comunità e/o direttamente da Microsoft.

Aggiungendo il supporto per un determinato linguaggio, come Python, VSC fornisce funzionalità di debugging, integrazione con il terminale e strumenti per la formattazione e l’esecuzione del codice, trasformandosi a tutti gli effetti in un IDE completo.

Jupyter, invece, è un ambiente di sviluppo interattivo che si basa sui cosiddetti “notebook”.

In un Jupyter Notebook, il codice è organizzato in celle che possono essere eseguite in modo indipendente.

Questo approccio facilita l’esplorazione dei dati e la realizzazione di prototipi, consentendo di alternare blocchi di codice e blocchi di testo descrittivo.

È particolarmente apprezzato da chi si occupa di data science e machine learning, perché permette di documentare e visualizzare i risultati in modo dinamico ed immediato.

Per iniziare a sviluppare con Python in Visual Studio Code o Jupyter, occorre innanzitutto installare l'interprete Python, che fornisce le fondamenta su cui basare il proprio ambiente di lavoro.

A seguire, è necessario configurare opportunamente il proprio strumento (VSC o Jupyter) per assicurarsi che riconosca e utilizzi correttamente l'interprete Python.

Di seguito trovi una lista di passaggi generali per procedere con l'installazione e la configurazione di Python in entrambi gli ambienti.

Lista dei passaggi per installare Python su VSC e Jupyter:
Scaricare e installare la versione più recente di Python dal sito ufficiale. Durante l'installazione su Windows, assicurarsi di selezionare l'opzione "Add Python to PATH".

Per Visual Studio Code:

- **Avviare VSC e installare l'estensione "Python" dal marketplace interno (icona a forma di quadratini sulla sidebar).**
- **Verificare che VSC riconosca la corretta versione di Python (in basso a destra, selezionare l'interprete).**
- **Facoltativo: installare eventuali estensioni aggiuntive per il debugging o il supporto a librerie specifiche.**

Per Jupyter:

- **Aprire il terminale o prompt dei comandi e digitare pip install jupyter oppure installare Anaconda, che include Jupyter di default.**
- **Avviare Jupyter Notebook (o JupyterLab) con il comando jupyter notebook o jupyter lab.**
- **Creare un nuovo notebook selezionando il kernel Python corretto.**

Eseguire un semplice test di stampa (ad esempio print("Hello, Python!")) per verificare che tutto sia configurato correttamente.

Python è un linguaggio di programmazione di alto livello, introdotto nel 1991 da Guido van Rossum.

È caratterizzato da una sintassi chiara e leggibile, che lo rende particolarmente adatto a chi si avvicina per la prima volta alla programmazione.

Python adotta un paradigma multi-paradigma, supportando la programmazione orientata agli oggetti, quella procedurale e, in parte, quella funzionale.

La filosofia del linguaggio si riassume nella volontà di favorire la semplicità e la produttività dello sviluppatore.

Sul piano tecnico, Python è interpretato e dinamicamente tipizzato: ciò significa che non è necessario dichiarare esplicitamente i tipi delle variabili e che l'esecuzione del codice avviene tramite un interprete (anziché essere compilata in un eseguibile).

Questa caratteristica semplifica notevolmente lo sviluppo, poiché consente di scrivere e testare il codice in modo più rapido.

Tuttavia, per applicazioni che richiedono altissime prestazioni, potrebbero essere preferibili linguaggi compilati (come C++ o Rust).

Python gode inoltre di un vastissimo ecosistema di librerie e framework, utili per un’ampia gamma di applicazioni: dalla data science (NumPy, Pandas, SciPy, scikit-learn) allo sviluppo web (Django, Flask), passando per machine learning, intelligenza artificiale e automazione di processi.

La ricchezza di risorse, unita alla solidità della community, rappresenta uno dei punti di forza principali di Python, che continua a crescere in popolarità e ad affermarsi in molteplici contesti industriali e accademici.

Ecco un elenco delle principali caratteristiche tecniche di Python, accompagnate da una breve spiegazione di ciascuna:

Linguaggio interpretato

Python non richiede una fase di compilazione: il codice viene eseguito direttamente da un interprete. Questo semplifica il ciclo di sviluppo, permettendo di testare le modifiche in modo immediato.

Tipizzazione dinamica

Non è necessario dichiarare esplicitamente il tipo di una variabile. Il tipo è infatti determinato automaticamente a runtime, rendendo il linguaggio più flessibile e facile da imparare, ma talvolta meno performante rispetto ai linguaggi a tipizzazione statica.

Multi-paradigma
Python supporta diversi stili di programmazione: orientato agli oggetti, procedurale e anche funzionale (sebbene in misura meno spinta rispetto a linguaggi dedicati). Questo permette di adottare lo stile più adatto al problema da risolvere.

Ampia libreria standard
Python mette a disposizione una libreria standard molto ricca, che include moduli per gestire file, connessioni di rete, processi, parsing di dati e molto altro. Questo riduce la necessità di librerie esterne per le funzionalità di base.

Memory management automatico
Python integra un garbage collector che si occupa di gestire la memoria liberando automaticamente lo spazio non più utilizzato. Ciò permette agli sviluppatori di concentrarsi sulla logica dell'applicazione, senza preoccuparsi dell'allocazione manuale della memoria.

Ecosistema e community estesi
Oltre alla libreria standard, esiste un vasto numero di librerie e framework di terze parti (NumPy, Pandas, Django, Flask, ecc.) sostenuti da una comunità molto attiva, che rende l'apprendimento e l'utilizzo di Python ancora più agevoli.

Portabilità e multiplatform
Essendo multipiattaforma, Python può essere eseguito su Windows, macOS, Linux e molte altre piattaforme. In questo modo, il codice è facilmente trasportabile da un sistema operativo all'altro.

Elevata leggibilità
La sintassi di Python è progettata per essere pulita e intuitiva. L'indentazione è parte integrante della struttura del linguaggio, contribuendo a rendere il codice più leggibile e semplice da mantenere.


Buon Davante a tutti