Language	Pros	Cons
Python	• Extremely popular • Lots of community packages • Lots of training material • Virtually all platforms and 32/64-bit • 100% open source • Actively developed and evolving language • Most taught language in colleges and universities (recently)	• No unified UI development environment • No native table/database layer • Dependency nightmare • No compilation error detection • No easy single packaging • Your code can easily be reverse-engineered • Indentation-based logic • One of the slowest in runtime • Really different approach to linking (importing) external modules • Case-sensitive language
Java	• Virtually all platforms and 32/64-bit • Lots of training material • Still popular	• No native table/database layer • Your code can easily be reverse-engineered (decompilation) • Case-sensitive language • For newer versions, commercial runtime license fees are required and could become expensive • OpenJDK is open source, but not the official implementation by Oracle
JavaScript (Node)	• Extremely popular • Lots of community packages • Lots of training material • Actively developed and evolving language • Very fast execution	• No native table/database layer • Dependency nightmare • No compilation error detection • Source code deployment • Web backend mainly, desktop solutions requires solution like electronjs • Case-sensitive language
C# and VB.NET	• Popular • Rich development environment • Lots of training material • Desktop and web solutions • Virtually all platforms and 32/64-bit with reduced functionality	• Controlled by single vendor, Microsoft (which terminated VB, Visual FoxPro) • No native table/database layer • Your code can easily be reverse-engineered (decompilation) • Overly complicated class structure and API • Case-sensitive language • Not 100% open source
PHP	• Easy syntax • Lots of training material • 100% open source • Still in the top 10 most popular languages	• No compilation error detection • Source code deployment • Web backend only, not for desktop solutions • On the decline in popularity • Case-sensitive language

Language	Pros	Cons
C++	• Fast Runtime • Virtually all platforms and 32/64-bit	• No native table/database layer • Extremely verbose language • Really difficult to debug • Case-sensitive language
Ruby	• Nice syntax	• On the decline in popularity • Web backend only, not for desktop solutions • Slow runtime • Case-sensitive language
xHabour	• Virtually the same syntax as Harbour, XBase++ and VFP • Virtually all platforms and 32/64-bit • Hard to reverse engineer (no practical decompilation) • Virtually the same syntax as Harbour, and VFP • Fast runtime • Multiple database engine native support, like DBF, Advantage Database (commercial) • Case insensitive language	• No visible active development • Single vendor • Harbour already integrated all of its features
XBase++	• Direct support from vendor (Alaska-Software) • Hard to reverse engineer (no practical decompilation) • Virtually the same syntax as Harbour, and VFP • Very good web protocol support • Fast runtime • Multiple database engine native support, like DBF, PostgreSQL, Advantage Database (commercial) • Case insensitive language	• Proprietary, not open source • Windows 32-bit only • Limited interoperability • Yearly license fee • Vaporware regarding their VFP support • Weak UI design tools
X#	• Possible solution for .NET developers to add xBase language syntax	• .NET runtime • Your code can easily be reverse-engineered (decompilation) • Case-sensitive language • Still under development • Commercial license may be required
VFP	• Built-in fantastic IDE, including forms designed with visual inheritance and report writer • Excellent for creating desktop Windows 32-bit apps • Very rich language, everything but the kitchen sink approach • Extremely stable • SQL syntax support on DBF	• End of Life by Microsoft • Your code can easily be reverse-engineered (decompilation) (unless commercial branding) • Windows 32-bit only, limited to 2GB table size • Closed code, and non-free development environment

Language	Pros	Cons
	• Extremely fast for table queries and inserts (was the foundation of the MS SQL 7+ engine) • In-memory tables (cursors) • Case insensitive language	
Harbour	• 100% Open Source and free • Virtually all platforms and 32/64-bit • Hard to reverse engineer (no practical decompilation) • Virtually the same syntax as xHarbour, Xbase++, X#, and VFP • Fast runtime • Multiple database engine native support, like DBF, SQLite, Advantage Database (commercial) • Case insensitive language • In 64-bit, will break the 2GB table limit • Support for in-memory tables • FastCGI framework for web development	• No vibrant core developer community • No unified UI development environment • Lack of clear language documentation (getting better) • Fragmented core code (branching) • No local table SQL syntax support • No easy build process