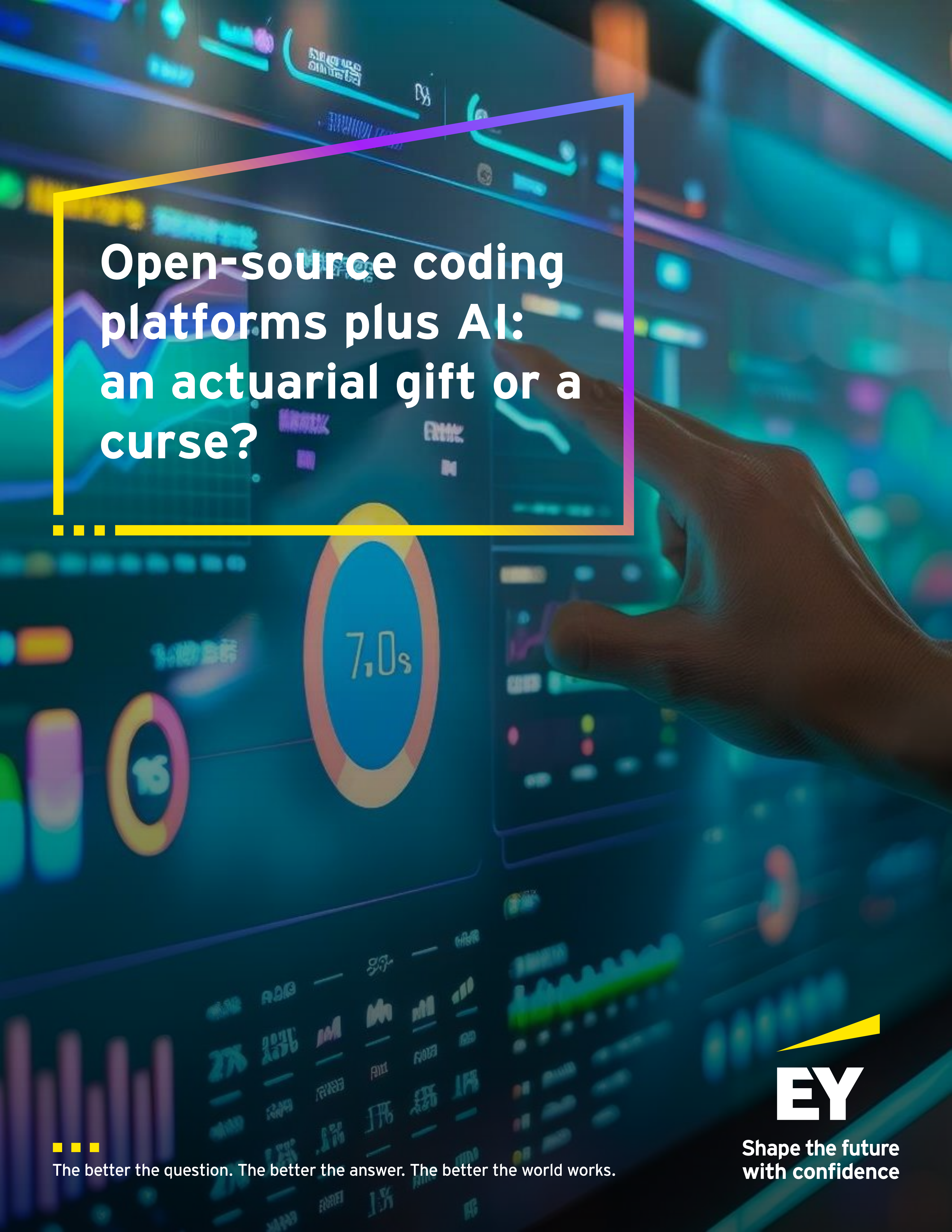

Open-source coding platforms plus AI: an actuarial gift or a curse?

The better the question. The better the answer. The better the world works.

EY

Shape the future with confidence

When artificial intelligence is enabled, will its combination with open-source coding platforms provide total value that is greater than the sum of its parts?

As technology has advanced over the years, so too have the toolsets that actuaries have at their disposal to accommodate increasingly complicated inquiries. This includes a mix of actuarial valuation and projection models, spreadsheets, databases, process automation and analytics applications. Fast-forward to today and we see many actuarial departments have modernized their technology stack in an effort to meet the challenges of recent accounting and regulatory changes, such as US GAAP long-duration targeted improvements (LDTI) and principles-based reserving. These modernization efforts have yielded consolidated financial modeling platforms that uniformly service both valuation and projection needs, curated and warehoused data capabilities, and automated processes with detective controls. These modernized toolsets not only have helped companies meet the challenges of accounting-related change but have also taken steps towards foundational advanced analytics capabilities to provide deeper business insights and respond more nimbly to the unforeseen.

Despite the benefits cited, challenges remain. Needs continue to emerge for the highly customized treatment of leading-edge product designs that are coupled with high "speed-to-market" demands. Older, outdated technologies, such as a programming language (APL) or spreadsheets, still persist despite the availability of superior alternatives. Runtime challenges remain omnipresent in more complicated use cases like asset-liability management (ALM) and forecasting and planning and analysis (FP&A), further adding to the cloud and distributed processing costs as demands continue to increase.

To address these challenges historically, actuarial teams would have faced onerous expenditures of energy, time and money to address these challenges. At EY, we've challenged our teams with the question, "what can we do differently this time?" , or "what can we do to leverage the rapid evolution in AI and compute capabilities, and maximize the human contribution?".

Recently, we have observed heightened interest in the market in two technologies that while individually provide a lot of promise, collectively present unique synergistic opportunities to enhance the actuarial toolset. Open-source programming languages, such as Python, present a simple, transparent and easy-to-use coding platform that has wide adoption and a plethora of readily available libraries with useful capabilities that can be easily leveraged. Artificial intelligence (AI), now ubiquitous in the business and technology "town square," offers powerful efficiency gains and insights that the business world is still in the process of harvesting. This article explores the tremendous power that the combination of Python and AI offers, the practical applications in the actuarial toolkit and considerations for adding them to your existing actuarial toolkit. For purposes of this article, we will focus on Python as a representative example of open-source programming languages but could be extended to other languages (such as R, C#, Julia, Java, etc.).

Getting to know Python

Drawing from its website, Python is described as "an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together."[1]

Python has gained popularity due to its relatively low learning curve, its large and active community of users and developers, and its versatility — particularly for analytics domains like data science and machine learning. Python also offers a wide range of ready-made libraries to accelerate development (more on this below).

To work effectively with Python, there are a few key components to consider

Basic installation

From its website (python.org), you can obtain and download the latest version to your relevant operating system of choice.

To work with the system, most users then install an integrated development environment (IDE), such as Visual Studio Code, PyCharm or IDLE, which provide features like syntax highlighting, debugging and code completion.

Library acquisition

The real power of Python can be found in its extensive ecosystem of libraries. A Python library contains preprogrammed functionality that can be easily integrated within Python scripts to allow for immediate usage and functionality. As of this writing, there are hundreds of thousands of Python libraries available for free and they can typically be installed via a simple command line statement.

A summary of some of the more popular and relevant Python libraries is provided below.

Library	Description	Actuarial applications
pandas dask modin	Generates "dataframes" (structured containers) The optimal package is chosen to optimize for different data challenges (ease of setup, size of data set, parallelization capabilities).	▪ Store input data and assumptions ▪ Store and manipulate modeled projection calculations
numpy	Enables fast computation and vectorization, optimized with C "under the hood"	▪ Any actuarial calculations, particularly where vectorization and speed are critical
sqlalchemy	Allows direct read and write access to database applications like SQLServer, mySQL and Snowflake	▪ Connect models to sourced input and output data ▪ Assist in the warehousing of assumption data
beautifulsoup	Enables web "scraping" to extract data from HTML and XML files	▪ Automated routines to pull enrichment data like interest rates and equity returns
flask	Provides a lightweight and flexible framework for building web applications and application programming interfaces (APIs)	▪ Actuarial process orchestration, triggering processes and tools via APIs
matplotlib seaborn	Libraries for data visualization that support static, animated and interactive plots	▪ Complement back-end reporting capabilities
scikit-learn	Powerful machine learning library for data preprocessing, classification, regression, clustering and model evaluation	▪ Predictive modeling and assumption setting
lifelib	Actuarial modeling library that includes sample scripts and Jupyter Notebooks to accelerate modeling efforts	▪ Model validation and testing ▪ Pricing and profit testing ▪ Valuation and cash flow projections

[1]"Python," Python Software Foundation website, https://www.python.org.

Other enablers

There are also a host of mostly free applications and sites that can enhance the coding experience. This includes Jupyter Notebooks for integrating code with real-time analytics and documentation (more on this later); GitHub for storing, sharing and collaborating on code; and copilot technologies (including GitHub CoPilot) that can provide real-time AI-enabled support for code development.

Benefits of Python

With these components in hand, Python offers benefits commensurate with the description shared above from its own website, including:

1 Speed to market

Quick setup capabilities via enablers like libraries

2 Low learning curve

Novice users learn Python quickly, leverage IT disciplines more easily

3 Lightweight

Ability to create targeted solutions that could present processing advantages

4 Controllable

Industrial-strength versioning toolsets (e.g., Git) support model development lifecycle activities

5 Conversant with other toolsets

Easy access to APIs, and other built-in plug-ins, making Python easily connectable to other upstream and downstream applications

Life insurers have commonly embraced one-off codesets or side models for very targeted purposes, often to attempt to fill a gap left by more general-purpose technologies. For many, this has involved forays into APL, Fortran, Visual Basic and Visual Basic for Applications, among others. The industry has since taken notice of Python's flexibility and power, and we are seeing increased exploration and usage of it — not only to replace outdated coding languages but also as a strategic tool for broader actuarial applications.

The benefits of applying AI to Python

Like most other industries, life insurers have been scrambling to find ways to harness the power of AI. Technologies like generative AI (GenAI) provide groundbreaking capabilities to review and interpret text and language with lightning speed.

Investments to-date have been largely directed to service operational demands; we see the proliferation of this technology to the actuarial domain leading to better business insights that improve forecasts, help efficiently manage capital, and ultimately drive earnings growth.

These AI capabilities have been rapidly evolving especially over the last two years, driven not only by the increasing sophistication of the models themselves, but also by the democratization of AI, which sees AI embedded in our daily lives. As AI has become more accessible and trusted, its adoption continues to expand — whether that's with smarter search engines, an AI copilot to book your vacation or a coding companion. The same can be said within the actuarial world where teams are using daily AI copilots both for more mundane tasks (e.g., meeting minutes) and for more complex activities like code development.

Integration of AI capabilities with open-source coding platforms like Python presents particularly strong synergistic opportunities, leveraging the open architecture nature of Python alongside AI's powerful ability to process and interpret text. The convergence of these technologies into common platforms has resulted in a Python coding experience that is incredibly efficient and optimized. Examples of this include:

Jump start code development	Interrogate a codeset with AI to gain quick insights	Help refining code
A common application of GenAI is to have it attempt to write Python code based on a series of inquiries. With some simple statements, solutions such as ChatGPT are able to write fulsome Python scripts to accommodate requests. While not perfect, it serves as a great accelerator to jump-start Python coding, especially for the novice user.	In lieu of reading model documentation, searching through code and making your own interpretations, a GenAI tool can be asked direct questions and return insights and summaries that provide direct answers. Because of the open architecture nature of Python, these interrogations are constrained by proprietary "walls" around codesets, which, if in place, can limit the effectiveness of the inquiries.	Many are familiar with "intellitext," a technology that will attempt to help you finish sentences as you type. Most coding platforms include this as well; in the case of Microsoft's VS Code, a new "sidecar" has been added called GitHub Copilot that enables interrogation and suggests changes that directly reference the code you are already writing. So in lieu of doing a web browser search, getting an answer and figuring out how to incorporate in your code, the GitHub Copilot will provide a contextualized answer to your inquiry and suggest direct modifications to your code. If that's not enough, it will also provide intellitext support that will do a lot more than just anticipate the completion of the current line of code.

The net result of this is a coding toolset that now can be constructed in a much more expeditious manner, with a lower learning curve and with substantially more transparency.

Opportunities for actuarial

While we recognized earlier the modernization efforts that have successfully enhanced actuarial capabilities, we feel there are additional opportunities for inclusion of this technology in the actuarial toolset. This technology could be leveraged both for bespoke use — as many have used tools like Visual Basic for Applications, Excel or others for years — as well as for more dedicated and targeted use cases, as noted below.

Use cases requiring a targeted solution

Some use cases require a great deal of detailed modeling while simultaneously retaining the ability to possess fast processing times in order to make the analysis actionable. Forecasting-oriented use cases come to mind, such as product pricing, capital planning and ALM, due to the level of detail and intensive runtime calculations. An open architecture coding solution empowered with AI can potentially provide:

- A more lightweight solution to the problem, having scope and code to focus just on the demands of the use case in question without needing to address others and be "all things to all people"
- An ability to spin up solutions quickly, with the enablement from AI tools for code generation and maintenance to meet emerging and unforeseen demands quickly
- More expedient runtimes, resulting in added speed to market and lower run production costs

Models with outdated codesets

Most insurance companies have selected utilities and models residing on out-of-favor codesets or platforms, such as APL, Cobol or Fortran. These utilities, while functional, can present eventual challenges such as key person risk or the prospect of the technology being sunset or becoming unsupported, among others. AI could help migrate these utilities and models into an open-source codeset that would build the foundation for future flexibility.

Expansion of research and development (R&D) insights

Python and its stable of libraries provide a powerful platform to explore actuarial research areas, such as experience analysis and runtime acceleration. A good example of this is in the predictive modeling space to support experience analysis and assumption setting. Establishing a Python environment, especially equipped with some of the statistical libraries mentioned earlier, provides ready access to a range of statistical models and analytics to support experience analysis and actuarial assumption determination. Seating the code within a Jupyter Notebook enables the user to directly integrate statistical analysis within the available codeset, providing real-time analytics as the actuary refines the analytics approaches. A unified environment and supporting GenAI methods facilitate streamlined documentation of the outcomes.

Increasing end-to-end connectivity

Most actuarial modeling processes are stitched together across a variety of technology platforms and process automation toolsets. Some toolsets are more conversant than others, presenting open APIs that can be connected to allow more seamless automation solutioning, whereas others require bespoke workarounds. While not a "silver bullet" for that issue, insertion of Python to the end-to-end process introduces a component that is very receptive to inter-application connectivity and can stand to help clean things up. We have seen insertion of Python beyond direct modeling to include automation of run execution, parsing run logs to provide end users with better and more immediate diagnostic information, and to also scrape external data sources to enrich data produced by the modeling process.

Beyond the specific practical applications listed above, there are other second-order opportunities to consider, such as addressing the evolution of the actuarial talent pool. The skill sets required to harness Python and AI cross over a bit more into the IT realm, which can create opportunities for actuarial departments to rethink their staffing. This can involve heavier involvement from IT to handle day-to-day modeling demands, potentially freeing up actuarial staff for more actuarial-focused tasks and value-add.

A “great power and great responsibility” situation

With any emerging technology solution, the initial — sometimes boundless — optimism one may feel must be checked by the associated risks and considerations of its adoption. In this case, the power is clearly available with the prospect of open architecture solutions for actuarial usage. But, as the saying goes, harnessing it comes with some degree of heightened responsibility and consideration for other effective resident technologies already in play, including:

Increased need for refined model development lifecycle (MDLC) and governance

Open architecture is just that — open to user modifications without constraints. Vendor-provided solutions tend to levy some constraints on the extent an individual can modify their code base in an effort to preserve the functionality and integrity of their products. The governance benefits of this approach must be appreciated and recognized as valuable features of these models. As such, a move to open-architecture systems such as Python will require more industrial strength refinement and enhancement of one’s MDLC techniques. This can include operating model refinements, including the type of talent acquired to maintain the models, and technology-based tools to help restrict user access and help manage changes to model files effectively.

Relationship with other models

Introduction of new models will create maintenance challenges with more general-purpose actuarial systems already in play. Inclusion of new products, regulatory requirements or analytics enhancements may bring the prospect of updating multiple times across models, depending on the scope of your open-architecture-side models. Consideration should be given to the scope of Python usage, how tightly the models must adhere to broader model governance guidelines and how tethered the models must be to methodologies embedded in general-purpose modeling applications.

Adapting talent pool skill sets

Actuaries are increasingly entering the workforce with open architecture coding skills gained from their experience at university.

Nonetheless, skill set demands will continue to evolve and the entrance of these toolsets can raise the bar for multidisciplinary talent, or drawing more heavily on IT resources. Furthermore, harnessing the power of AI will require a staffing model that understands how to do so, while at the same time promoting safeguards and practices to secure company data and associated trade secrets.

What the future holds

We mentioned at the outset there have been tremendous strides made by the industry in the transformation agenda to meet the challenges of accounting changes, plenty of open-source technologies and AI are offering a catalyst to explore the next generation of enhancements to the actuarial process, with model development being a strong use case. The industry will continue exploring ways to monetize AI for its benefit, and it is important for actuarial departmental research to keep pace as the landscape changes quickly. We see a strong example showing how AI, when combined with another trusted application, like Python, can create a synergy where the combined value has the prospect of being greater than the sum of its parts.

Ernst & Young LLP contacts

Dave Czernicki
EY US Actuarial Modeling Leader
dave.czernicki@ey.com

Eric Wolfe
Managing Director
eric.wolfe@ey.com

Sian Walker
Manager
sian.walker1@ey.com

EY | Building a better working world

EY is building a better working world by creating new value for clients, people, society and the planet, while building trust in capital markets.

Enabled by data, AI and advanced technology, EY teams help clients shape the future with confidence and develop answers for the most pressing issues of today and tomorrow.

EY teams work across a full spectrum of services in assurance, consulting, tax, strategy and transactions. Fueled by sector insights, a globally connected, multi-disciplinary network and diverse ecosystem partners, EY teams can provide services in more than 150 countries and territories.

All in to shape the future with confidence.

2504-10680-CS
US SCORE no. 26898-251US
ED None

ey.com