Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Algorithmic Stratification in Finance Degrees: Global Patterns of Python Adoption in University Curricula

Veliota Drakopoulou

vdrakopoulou@yahoo.com

Higher Colleges of Technology; Embry Riddle Aeronautical Univesrity

Research Article

Keywords: algorithmic literacy, finance education, Python, financial technology, management education, curriculum design, higher education stratification, labor-market outcomes, computational skills

Posted Date: February 24th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-8937498/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: The authors declare no competing interests.

Algorithmic Stratification in Finance Degrees: Global Patterns of Python Adoption in University Curricula

Veliota Drakopoulou
[1] Higher Colleges of Technology
[2] Embry-Riddle Aeronautical University
ORCID: 0000-0002-1670-8033
Correspondence to: vdrakopoulou@yahoo.com
https://github.com/vdrakopoulou
Replication Package: DOI: 10.5281/zenodo.18713524

Abstract

Universities and business schools are under growing pressure to equip finance graduates with the algorithmic literacy required for data-intensive, Python-based work in quantitative finance and financial technology, yet little is known about how far Python has been institutionalized in finance curricula, and for whom. This research conceptualizes Python as a form of curricular infrastructure and an axis of algorithmic stratification and provides the first systematic cross-national map of its documented presence in finance degrees. Using publicly available programme and module descriptions, the study codes 241 finance-related programmes offered by 140 universities in 14 countries. A three-dimensional rubric captures the depth, scope and authenticity of Python integration and is combined into a Python adoption index (0–12; normalized 0–1). Cross-national and programme-level patterns are examined with descriptive statistics, analysis of variance and regression, complemented by mixed-effects models with random intercepts for country and thematic analysis of programme-level comments.

The results reveal a highly stratified landscape: around one third of programmes document no Python, another third show limited or elective-only provision, and a smaller group exhibit deep, authentic, programme-wide integration. More than half of the variance in the adoption index lies between countries, and Python is strongly concentrated in graduate and specialist quantitative/FinTech programmes. Among programmes that do adopt Python, however, the intensity of integration converges across degree levels.

Empirically, the study offers the first systematic cross-national map of Python integration in finance education; methodologically, it develops a transferable Depth–Scope–Authenticity framework; and conceptually, it demonstrates how Python structures access to algorithmically intensive segments of the finance labor market, with implications for curriculum design, professional standards, and equity in higher education and management education.

Keywords

algorithmic literacy; finance education; Python; financial technology; management education; curriculum design; higher education stratification; labor-market outcomes; computational skills

1. Introduction

Finance has been reshaped by digitalization and data-intensive technologies. Trading, risk management, pricing and regulatory reporting are now tightly coupled to software infrastructures and data pipelines, with languages such as Python increasingly embedded in the everyday tools of quantitative finance and financial technology (Hilpisch, 2018; MacKenzie, 2017; McKinney, 2018; Poon & Wójcik, 2019). In this context, the ability to work with code and data has become an important component of professional competence in many segments of the financial sector, alongside more traditional knowledge of markets, instruments and institutions (Autor, 2014; Deming, 2017).

Labor-market and professional signals reinforce this shift. Employer surveys and occupational forecasts highlight programming and data-analytic skills as core requirements for high-growth roles in financial analysis, risk, and FinTech, frequently naming Python as a preferred language (CFA Institute, 2020; U.S. Bureau of Labor Statistics [BLS], 2023; World Economic Forum [WEF], 2020). Professional associations have begun to integrate data analytics and basic programming concepts into their curricula and guidance, and universities are under increasing pressure to demonstrate that graduates are prepared for data-intensive work in finance and related fields (Brown, Lauder, & Cheung, 2020; CFA Institute, 2020).

Despite this, there is limited systematic evidence on how Python has been incorporated into university-level finance education. Existing studies of the "computational turn" in higher education tend to focus on computer science, engineering and generic business analytics, with finance appearing only in single-programme case studies or brief examples (Arthurs & Walsworth, 2013; Brown et al., 2020; Grosemans & Notebaert, 2022). Where programming is reported in business or finance curricula, different languages and uses are often collapsed into a single indicator, and there is little cross-national analysis of how integration varies between systems, institutions, degree levels and programme types. As a result, it is not clear where Python appears in finance curricula, how intensively it is used, or how its presence maps onto broader patterns of differentiation in higher education.

This article addresses these gaps by analyzing a cross-national sample of 241 finance-related programmes offered by 140 universities in 14 countries, using programme and module descriptions to track the documented presence of Python in the formal curriculum. A three-dimensional rubric is developed to capture the depth, scope and authenticity of Python integration and combined into a Python adoption index. The analysis focuses on four questions: (a) how prevalent Python is in finance-related programmes across different national systems; (b) how programmes differ in the intensity and character of integration; (c) how integration is patterned by degree level and programme type, particularly the distinction between undergraduate versus graduate and general finance versus specialist quantitative/FinTech degrees; and (d) how institutions discursively position Python in their programme-level descriptions. The focus is on the documented curriculum, programme webpages, online handbooks and catalogues which constitute the primary interface through which students, regulators and employers encounter programmes (Biggs, 1996; Bowen, 2009; Neuendorf, 2017).

The research makes three main contributions. Empirically, it provides the first systematic cross-national mapping of Python integration in finance curricula, documenting the extent of Python's presence, the distribution of programmes with no documented Python and the concentration of intensive provision in particular systems and programme types. Methodologically, it introduces a transferable Depth–Scope–Authenticity framework and associated index that enable more nuanced measurement of programming integration than simple presence–absence indicators, and that can be adapted to other languages and domains. Conceptually, the analysis links observed patterns of Python integration to debates on digital skills and stratification in higher education, arguing that Python now functions as a curricular element through which access to data-intensive segments of the finance labor market is structured.

The remainder of the article is organized as follows. Section 2 reviews literature on algorithmic literacy, the computational turn in finance education, labor-market and professional signals, and stratification in higher education, and develops a conceptual framework for understanding Python as curricular infrastructure. Section 3 outlines research design, sampling, coding rubric and analytic strategy. Section 4 presents empirical results. Section 5 discusses their implications for cross-national patterns of curriculum reform, programme-level stratification and equity in access to algorithmic skills, and Section 6 concludes with limitations and directions for future research.

2. Literature Review and Conceptual Framework

2.1. Algorithmic literacy and finance work

The notion of *algorithmic literacy* has emerged at the intersection of critical data studies, sociology of work and education research to describe the capacity to understand, use and critically interrogate algorithmic systems (Boyd & Crawford, 2012; D'Ignazio & Klein, 2020; Kitchin, 2017). In contrast to generic "digital skills," algorithmic literacy emphasizes the ability to translate domain problems into computational form, implement and adjust models in code, interpret outputs in context and reflect on the assumptions, biases and limitations embedded in data and algorithms (D'Ignazio & Klein, 2020; Kitchin, 2017; Seaver, 2017).

Finance is a paradigmatic site for the rise of algorithmic work. Over recent decades, trading, risk management, asset pricing and portfolio construction have become deeply entangled with quantitative models and large-scale software systems (MacKenzie, 2017; Poon & Wójcik, 2019). High-frequency and algorithmic trading, complex derivatives pricing, stress testing and regulatory reporting increasingly depend on code-based infrastructures and data pipelines, often mediated through languages such as Python, R, C++ and domain-specific platforms (Dörry & Pike, 2021; Godechot, 2016; MacKenzie, 2017).

Within this computational environment, finance professionals are expected not only to consume model outputs but also to interact directly with code and data, whether by prototyping strategies, querying databases, building risk dashboards or debugging scripts (Arthurs & Walsworth, 2013; Deming, 2017). Algorithmic literacy in finance therefore encompasses at least three dimensions. A syntactic dimension involves familiarity with programming constructs, libraries and tools. A semantic dimension involves understanding the financial and statistical meaning of code and outputs, for example, the implications of a particular volatility model, regression specification or

machine learning pipeline (MacKenzie, 2017; O'Neil, 2016). A critical dimension involves recognizing how model design, data choices and institutional incentives shape outcomes, including issues of opacity, fairness and systemic risk (Boyd & Crawford, 2012; D'Ignazio & Klein, 2020; Fourcade & Healy, 2017).

The literature suggests that algorithmic literacy has become a key component of occupational competence in many segments of finance. Graduates lacking such literacy may be confined to roles with limited autonomy and progression, even when possessing strong traditional finance knowledge, whereas those who can navigate models and code may access the more prestigious, better-remunerated "quantitative" and FinTech segments of the field (Autor, 2014; Godechot, 2016; Nolan, 2021).

2.2. Python and the computational turn in finance education

In parallel with these developments in practice, a growing body of work has examined the "computational turn" in higher education, particularly in STEM, data science and business education (Brown, Lauder, & Cheung, 2020; Deming, 2017; WEF, 2020). Studies of engineering and computer science programmes document the integration of project-based coding, data-intensive assignments and industry-linked capstones, often framed as responses to employer demand for "work-ready" graduates with strong programming skills (Freeman et al., 2014; Prince, 2004). Business and management programmes have similarly begun to incorporate analytics, data visualization and coding components, although often in elective or specialized tracks rather than across core curricula (Arthurs & Walsworth, 2013; Grosemans & Notebaert, 2022; WEF, 2020).

Python occupies a distinctive place within this computational turn. Its relatively simple syntax, extensive scientific libraries and open-source ecosystem have made it the dominant language in data science and an increasingly important tool in quantitative finance, where it is used for time-series modelling, derivatives pricing, risk analytics and algorithmic trading (Hilpisch, 2018; McKinney, 2018; O'Neil, 2016). Practitioner-oriented texts and online courses in "Python for finance" have proliferated, and industry surveys routinely list Python as a core language for quantitative and analytics roles (CFA Institute, 2020; WEF, 2020).

Evidence on how far Python has been diffused into *formal* finance curricula is more limited. Case studies describe individual courses or programmes that use Python to teach quantitative finance, econometrics or FinTech (e.g., Hilpisch, 2018; MacKenzie, 2017), and institutional reports highlight the introduction of analytics and programming streams in business schools (Arthurs & Walsworth, 2013; Grosemans & Notebaert, 2022). However, much of this literature focuses on isolated modules or local innovations and rarely distinguishes between superficial exposure (e.g., demonstrations or short workshops) and deep, programme-wide integration. Nor does it systematically compare Python's curricular presence across national systems or degree levels.

This gap matters because partial integration such as a single elective or a short bootcamp may do little to cultivate the sustained algorithmic literacy described in Section 2.1, especially if not reinforced in core finance modules or authentic assessments (D'Ignazio & Klein, 2020; Freeman et al., 2014; Prince, 2004). Understanding how Python is positioned within finance curricula therefore requires attention to *where* it appears (scope), *how intensively* it is used (depth) and *in*

what kinds of tasks (authenticity), as well as to cross-national and programme-level variation in these patterns.

2.3. Labor-market signals and professional standards (BLS, WEF, CFA Institute)

Labor-market and professional signals have played a central role in elevating programming and data-analytic skills in finance. Occupational projections and skills analyses consistently highlight rising demand for quantitative, analytical and technological competencies in financial services, especially in roles related to risk modelling, quantitative analysis, financial engineering and FinTech (Autor, 2014; Deming, 2017; U.S. Bureau of Labor Statistics [BLS], 2023). BLS projections for financial analyst, financial risk specialist and data scientist occupations emphasize the importance of statistical and programming skills, reflecting a broader shift towards data-intensive work (BLS, 2023).

Global policy reports reinforce this picture. The World Economic Forum's *Future of Jobs* surveys identify "data analysts and scientists," "AI and machine learning specialists," and "big data specialists" as rapidly growing roles in financial services, with programming, statistical analysis and data management listed as core skills (World Economic Forum [WEF], 2020). These reports frequently highlight Python, R and SQL as key tools in finance and related sectors, framing them as necessary complements to more traditional financial and business expertise (WEF, 2020).

Professional associations have begun to codify these expectations. The CFA Institute has progressively incorporated programming, data analytics and machine learning into its curriculum and has developed resources that explicitly recommend Python for quantitative topics (CFA Institute, 2020). Continuing education offerings and applied certificates increasingly foreground coding and analytics, and exam blueprints signal that future charter holders are expected to be conversant with data-driven methods and associated toolchains (CFA Institute, 2020; WEF, 2020).

From a curriculum perspective, such signals can be understood as part of the broader "marketisation" of higher education, in which programmes are pressured to align with employer demands and to highlight employability-enhancing skills in their marketing and design (Brown et al., 2020; Slaughter & Rhoades, 2004). If employers and professional bodies explicitly value Python-based algorithmic literacy, programmes that integrate Python deeply and authentically may confer a competitive advantage to their graduates, while those that do not may be perceived as outdated or misaligned with industry expectations. At the same time, the translation of labor-market signals into curricular change is mediated by institutional capacity, disciplinary cultures and national regulatory frameworks, leading to uneven uptake across systems and programme types (Altbach et al., 2009; Marginson, 2016).

2.4. Stratification in higher education and skills

The uneven integration of Python into finance curricula is likely to intersect with well-documented patterns of stratification in higher education. Sociological research shows that access to high-status institutions and programmes is structured by socio-economic background, prior attainment and geography, and that such institutions concentrate valuable forms of capital intellectual, social and cultural that translate into labor-market advantages (Bourdieu, 1984, 1988; Brown et al., 2020;

Marginson, 2016). The expansion and massification of higher education have not eliminated these inequalities; rather, they have often led to "effectively maintained inequality," in which advantaged groups secure access to more prestigious institutions, subjects and pedagogies within enlarged systems (Lucas, 2001; Marginson, 2016; Trow, 1974).

Within systems, stratification operates along vertical and horizontal axes. Vertically, elite research-intensive universities tend to offer more resources, smaller classes and greater access to cutting-edge content, while mass institutions deliver more standardized and resource-constrained curricula (Altbach et al., 2009; Slaughter & Rhoades, 2004). Horizontally, fields and subfields differ in their labor-market returns and prestige; finance, quantitative disciplines and STEM fields often yield higher earnings and status than many humanities and some social sciences (Autor, 2014; Gerber & Cheung, 2008; Marginson, 2016). Within finance itself, qualitative ethnographies and quantitative analyses point to a further segmentation between "front office" roles in investment banking, trading and asset management, and less remunerated roles in back-office operations or retail banking, with elite educational credentials serving as key filters (Godechot, 2016; Ho, 2009; MacKenzie, 2017).

Recent work on digitalization and the "opportunity hoarding" of high-level skills suggests that advanced computational competences are increasingly central to these processes (Brown et al., 2020; Fourcade & Healy, 2017; Nolan, 2021). Access to high-quality training in coding, data science and AI is often concentrated in elite institutions and programmes, while less advantaged students are more likely to encounter superficial or fragmented digital provision, if any (Brown et al., 2020; Grosemans & Notebaert, 2022; Ilieva & Killingley, 2022). In Tilly's (1998) terms, this can be seen as a form of "opportunity hoarding," in which privileged groups monopolize access to resources such as , algorithmic literacy that are increasingly valuable in the labor market.

Against this backdrop, the integration of Python into finance curricula is unlikely to be neutral. If Python-rich programmes are concentrated at elite institutions and in specialist quantitative degrees, while mass undergraduate finance programmes and institutions in less resourced systems remain Python-poor, algorithmic literacy is likely to become a new axis along which educational and occupational inequalities are structured (Fourcade & Healy, 2017; Nolan, 2021; Weeden, 2002).

2.5. Conceptual model: Python as curricular infrastructure and axis of algorithmic stratification

The literature suggests a conceptualization of Python as both a technical tool and a form of curricular infrastructure that underpins new patterns of algorithmic stratification in finance education and work. Drawing on science and technology studies, infrastructure can be defined as the often-invisible systems, standards and practices that enable and shape everyday activities (Bowker & Star, 1999; Star & Ruhleder, 1996). In educational contexts, infrastructural elements include not only physical technologies but also curricular structures, assessment regimes and institutionalized skill expectations (Biggs, 1996; Bowker & Star, 1999).

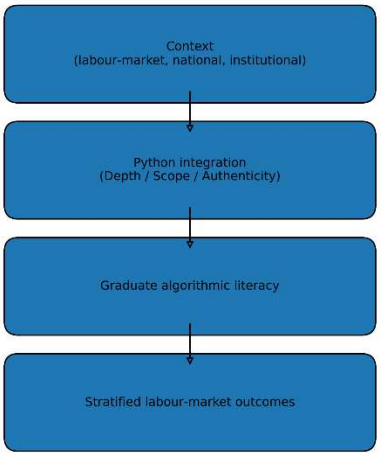
When Python is embedded deeply and authentically in finance curricula through required modules, programme-wide projects and explicit learning outcomes it becomes part of the taken-for-granted infrastructure through which students learn to "do" finance: accessing data, implementing models,

testing strategies and communicating results (Hilpisch, 2018; MacKenzie, 2017). Conversely, when Python is absent, relegated to electives or left implicit, the infrastructural backbone of finance education remains rooted in more traditional tools such as spreadsheets, calculators and purely analytical derivations. In this sense, decisions about Python integration are not merely technical but epistemic and social, shaping what counts as legitimate knowledge and practice in finance (Godechot, 2016; Ho, 2009; MacKenzie, 2017).

Figure 1 summarizes this conceptual framework. At the macro level, national context regulatory frameworks, funding regimes, labor-market structures and the strength of local financial centers create varying incentives and capacities for integrating Python into finance education (Altbach et al., 2009; Marginson, 2016; WEF, 2020). At the meso level, institutional factors such as mission, prestige, resources and disciplinary cultures influence whether Python is adopted, in which programmes and with what intensity (Brown et al., 2020; Slaughter & Rhoades, 2004). At the micro level, programme- and module-level decisions about curriculum design, assessment and pedagogy determine the depth, scope and authenticity of students' engagement with Python (Biggs, 1996; Freeman et al., 2014; Prince, 2004).

These curricular configurations shape graduates' algorithmic literacy syntactic, semantic and critical and thereby their positioning in a labor market that increasingly rewards such literacy in high-status finance roles (Autor, 2014; CFA Institute, 2020; WEF, 2020). The distribution of Python-rich and Python-poor programmes across institutions and countries thus becomes one mechanism through which algorithmic stratification and opportunity hoarding operate (Fourcade & Healy, 2017; Nolan, 2021; Tilly, 1998). In this framework, Python is conceptualized as an axis of *algorithmic stratification*: a curricular element whose presence, absence and form of integration structure differential access to the computational competences that underpin emerging segments of finance work. The empirical sections that follow use the Depth–Scope–Authenticity rubric and Python adoption index to operationalize this axis and to map its distribution across 14 countries, 140 universities and 241 finance-related programmes, thereby providing an empirical test of the conceptual model in **Figure 1.**

Figure 1. Conceptual framework: From context to Python integration and stratification



3. Methodology and Data

3.1 Research design and scope

The study adopts a cross-national, document-based curriculum analysis to map how Python is integrated into university-level finance education. The programme, rather than the individual module, functions as the primary unit of analysis. This focus follows curriculum research that treats official documentation as a structured statement of the "intended" curriculum, even if it does not fully capture classroom practice (Biggs, 1996; Young, 2013). Accordingly, the analysis relies on programme webpages, handbooks, regulations and course catalogues, supplemented where available by detailed module descriptions.

The design is cross-sectional and descriptive-analytic. A structured dataset is constructed in which each programme is scored on a three-dimensional rubric capturing the depth, scope and authenticity of Python integration, alongside programme-level metadata such as country, degree level, degree label, academic year and evidence type. The empirical strategy proceeds in three stages: first, cross-national patterns in a normalized Python adoption index are described; second, stratification by programme level and type is examined, with particular emphasis on contrasts between undergraduate and graduate programmes; and third, ordinary least squares (OLS) and mixed-effects models are estimated to assess whether graduate programmes exhibit systematically higher levels of Python integration, accounting for clustering by country.

The analysis is explicitly concerned with the documented curriculum and thus with how institutions represent finance education to prospective students, regulators and employers (Bowen, 2009; Neuendorf, 2017). Programme-level narrative comments are used to retain some of the contextual nuance that would otherwise be lost in a purely numeric coding scheme (Miles et al., 2014). All analyses are fully reproducible from the replication package, which includes the scored dataset, analysis scripts and machine-readable tables and figures, consistent with best practice in open and reproducible quantitative research (Peng, 2011).

The measurement strategy also builds on design principles developed in prior work on public signals of Python-enabled AI in finance. Drakopoulou (2025) maps disclosure patterns for NYSE-listed institutions using a strictly defined population frame, multi-channel evidence and transparent coding rules that treat public documentation as an observable but selective proxy for underlying computational capability. In that framework, firms are classified into distinct disclosure states and named tools are linked to functional domains, with the aim of producing a reproducible, population-level map of where Python and AI are explicitly acknowledged rather than an exhaustive census of internal practice. The present study extends this signal-based logic to higher education by treating programme-level documents as institutionally sanctioned artifacts that signal the formal status of Python in finance curricula and by using the Depth–Scope–Authenticity rubric to distinguish simple mention from sustained, assessed integration.

3.2 Country selection and university sampling (14 countries, 140 universities)

Country selection follows a maximum-variation strategy designed to capture diversity in both higher education systems and financial labor markets, within the practical constraints of a manually coded, document-based study. Fourteen countries were included to span: (a) large, advanced economies; (b) major global or regional financial centers; (c) large emerging markets with expanding capital markets and FinTech sectors; and (d) African systems positioned at different points in the global financial hierarchy. This approach is consistent with work that conceptualizes higher education and professional formation as embedded in stratified national and transnational fields (Marginson, 2016).

Within each of the 14 systems, universities were identified that (i) clearly offer at least one finance-related programme at the undergraduate or graduate level and (ii) provide sufficiently detailed online documentation to support application of the rubric. Institutional websites, business school pages, and course catalogues constituted the primary sources for this identification step. Sampling sought to include both research-intensive and teaching-oriented institutions, as well as both public and private providers where these distinctions are salient in the national context (Altbach, Reisberg, & Rumbley, 2009).

In total, 140 universities met these criteria and entered the sample. Appendix A (**Table A1**) reports the number of universities per country and illustrates that the sample spans both highly resourced and more resource-constrained systems. These institutions form the frame within which programme-level data on Python integration are collected.

3.3 Programme identification and inclusion criteria (241 programmes)

Programme identification proceeded in three stages. First, for each sampled university, institutional and faculty websites and course catalogues were searched for programmes whose titles or descriptions indicated a substantial finance component. Programmes explicitly labelled finance, financial economics, financial engineering, quantitative finance or financial technology were included, as were degrees where finance appears as a defined major, track or specialization within a broader programme (for example, an economics and finance degree or an MBA with a finance concentration). This approach reflects the view that algorithmic literacy in finance can develop within multiple formal degree structures (Cunningham & Vilaseca, 2016).

Second, identified programmes were screened against three inclusion criteria. Each programme had to lead to a recognized post-secondary credential (bachelor's, master's or formally designated professional degree), present a clearly delineated finance curriculum and provide sufficient documentation in English or another language accessible to the author to permit reliable coding of Python integration. Programmes described only through highly abbreviated marketing blurbs without substantive curricular detail were excluded, in line with recommendations for document-based content analysis (Neuendorf, 2017).

Third, where universities offered multiple eligible programmes, the aim was not to collapse these into a single institutional record but rather to retain them as distinct, nested units. Consequently, the final dataset contains 241 finance-related programmes nested within 140 universities. This structure allows examination of both cross-institutional and within-institution differences in Python integration, while recognizing that some universities maintain differentiated portfolios of generalist and specialist finance degrees (Teichler, 2004). **Appendix A (Table A2)** summarizes the distribution of programmes by degree level and broad region.

3.4 Document collection and programme-level comment writing

For each of the 241 programmes, an effort was made to assemble a comprehensive corpus of official documentation relevant to Python integration. Programme overview pages, formal regulations, handbooks, calendars and detailed module descriptors were collected wherever possible. When multiple versions of a document existed, preference was given to the most recent version explicitly linked to the 2024–2025 academic year (or the nearest available year), and both academic year and date of access were recorded, consistent with best practice in documentary research (Bowen, 2009). URLs and local file names were stored in the dataset to create an audit trail.

All collected documents were read in full. References to Python were identified, along with related programming content such as "programming in Python", "Python for finance", "Python labs" or references to a "programming language" accompanied by examples. For each programme, a short narrative comment was written that summarized how Python features in the curriculum, whether use is compulsory or elective, what forms of assessment are linked to Python and how Python is positioned relative to other tools such as R, MATLAB, C++ and Excel. Ambiguities, such as tool-agnostic references to "appropriate software" or "a suitable programming environment", were also noted.

These narrative comments serve dual functions. Substantively, they provide source material for the thematic analysis of how programmes position Python, allowing an examination of discourse as well as structure (Braun & Clarke, 2006; Miles et al., 2014). Methodologically, they document the reasoning behind the assigned rubric scores and thereby enhance transparency and replicability, as recommended in content-analytic research (Neuendorf, 2017). The comments are included in the replication dataset, subject to minor anonymization where necessary.

3.5 Measures

3.5.1 Depth, scope and authenticity rubric (0–4 each)

Python integration is operationalized through a three-dimensional rubric capturing depth, scope and authenticity, each scored on an integer scale from 0 to 4. The rubric is described in full in **Appendix B1**, with detailed examples in **Appendix B2**. It is designed to move beyond binary presence/absence indicators and to capture differences in the intensity and educational quality of Python use (Carifio & Perla, 2008).

Depth represents the vertical integration of Python in the curriculum. Depth = 0 is assigned when Python is not explicitly documented in any finance-relevant component of the programme. Depth = 1 corresponds to isolated use, typically a single elective module, short workshop or optional thesis-support activity. Depth = 2 indicates that at least one structured Python or "Python in finance" module exists but follow on use in other finance modules is limited. Depth = 3 denotes a curriculum where a core programming or Python course is followed by several finance modules that explicitly build on Python skills. Depth = 4 characterizes programmes where Python is embedded across the degree, with evidence of use at introductory, intermediate and advanced levels.

Scope captures the horizontal spread of Python across modules and learning contexts. Scope = 0 indicates no explicit use in finance modules. Scope = 1 is used where Python appears in a single module or highly circumscribed activity. Scope = 2 reflects concentration of Python within a recognizable cluster, such as a FinTech or data analytics stream. Scope = 3 applies when Python is used in several distinct subfields of finance, for instance econometrics, risk management, derivatives and trading. Scope = 4 is reserved for programmes in which Python is described as the primary or dominant implementation language across a wide swathe of the finance curriculum.

Authenticity refers to the extent to which Python is used in realistic, domain-specific tasks and assessments. Authenticity = 0 is assigned when documentation mentions only generic software use or theoretical coverage without assessed coding. Authenticity = 1 describes toy examples, demonstrations or non-assessed exercises with minimal engagement in realistic tasks. Authenticity = 2 corresponds to assessed labs or exercises using Python with simplified or generic datasets. Authenticity = 3 indicates assessed projects or examinations involving Python and real financial data, for example pricing options, constructing portfolios or analyzing returns. Authenticity = 4 refers to extended, project-based or externally facing work, such as capstone projects, industry collaborations or research dissertations in which Python plays a central role.

Programmes with no explicit reference to Python in any collected document are coded 0 on all three rubric dimensions. Although the scales are ordinal, it is common practice to treat such ratings as approximately interval for the purposes of scale construction and index formation, especially when supported by psychometric diagnostics (Carifio & Perla, 2008; Kline, 2016).

3.5.2 Python adoption index (0–12) and normalized index (0–1)

To summarize Python integration at the programme level, the three rubric scores are aggregated into a simple additive Python adoption index. For programme i, the index is defined as

$$\text{Python_Index}_i = \text{Depth}_i + \text{Scope}_i + \text{Authenticity}_i,$$

yielding a potential range from 0 (no documented Python) to 12 (maximum depth, scope and authenticity). The index embodies an equal-weight assumption in which each dimension is treated as an equally important facet of a broader construct of Python-based algorithmic literacy in finance. Equal-weight composites of conceptually parallel indicators are widely used in educational and social measurement, if indicators show adequate internal consistency (Bollen, 1989; Kline, 2016).

For comparability across figures and models, a normalized version of the index is also used, obtained by dividing by 12 so that values lie in the interval [0, 1]:

$$\text{Python_Index_Normalised}_i = \frac{\text{Python_Index}_i}{12}.$$

Internal consistency of the three-item scale is evaluated using Cronbach's alpha (Cronbach, 1951), and a one-factor model is estimated from the interitem correlation matrix to examine unidimensionality (Kline, 2016). The resulting alpha coefficient and factor loadings are reported in Section 4.4 and **Appendix C2**. Descriptive statistics for the component scores and both index variants appear in **Appendix C1**.

3.5.3 Programme-level covariates (degree level, graduate dummy, country)

Several programme-level covariates are included to support descriptive and inferential analysis. Degree level is recorded as a categorical variable based on the verbatim degree label in institutional documentation, such as BSc, BA, BBA, MFin, MSc, MBA or other national degree designations. These labels are summarized in **Appendix A2**.

To distinguish undergraduate from graduate provision in a way that is comparable across systems, degree labels are recoded into a binary graduate_program indicator. Programmes labelled BSc, BA, BBA, BSBA, BCom, B.Econ, MBM, MBus and explicitly designated undergraduate disciplines or MA (Hons) degrees are coded as undergraduate (0). All other programmes, including taught master's, MBAs, professional masters and advanced specializations presuming a first degree, are coded as graduate (1). This recoding aligns with standard distinctions in the comparative higher education literature (Teichler, 2004; UNESCO, 2015).

Country is recorded as a categorical variable indicating the nation in which the university is located. This variable is used descriptively in the analysis of variance (ANOVA) and serves as the grouping factor for random intercepts in mixed-effects models. In addition, the dataset includes a Python_Found flag, coded "Y" when Python is explicitly named in documentation and otherwise left blank. This indicator is used to define a subset of programmes with explicit Python adoption for robustness checks. Metadata fields record the academic year to which the documentation refers, the primary evidence type (for example, handbook, catalogue, webpage) and the date of access, providing contextual information for the coding decisions (Bowen, 2009).

3.5.4 Qualitative positioning themes from comments

The programme-level narrative comments form the basis for a qualitative analysis of how programmes discursively position Python within their finance offerings. The analysis follows an inductive thematic approach consistent with Braun and Clarke's (2006) framework. In an initial coding cycle, the comments were read closely and short codes were assigned to phrases that described Python's status (for example, core infrastructure, elective add-on, preparatory bootcamp), its relationship to other languages (for example, transition from MATLAB to Python, multi-language ecosystems) and its labor-market framing (for example, industry-standard, employability skill, digital capability).

In a second cycle, these initial codes were grouped into a smaller set of higher-order themes that captured recurrent positioning patterns, such as "Python as curricular backbone", "Python as optional enhancement", "implicit or non-documented Python" and "tool-agnostic quantitative training". A final pass through the data applied this refined codebook systematically to all comments, allowing multiple themes to be assigned to a single programme where appropriate. The resulting binary indicators for each theme are tabulated by country and degree level in **Appendix C3** and summarized in Section 4.7. This qualitative layer is intended to situate the quantitative indices within the broader narratives through which institutions present computational finance education (Miles et al., 2014).

3.6 Analytic strategy (descriptives, ANOVA, reliability, OLS, mixed effects, robustness)

The analytic strategy combines descriptive statistics, psychometric diagnostics and multilevel regression modelling, supplemented by robustness checks. All analyses are implemented in Python (version 3.x), using pandas and NumPy for data management, statsmodels for regression and mixed-effects models, SciPy for inferential tests and matplotlib for visualization. The complete analysis is scripted so that all tables and figures can be reproduced directly from the scored dataset, in line with reproducible research principles (Peng, 2011).

Descriptive statistics, including counts, means, standard deviations, minima, maxima and quartiles, are computed for Depth, Scope, Authenticity, the raw Python adoption index and the normalized index. These are reported in **Appendix C1** and provide a baseline description of the distribution of Python integration across programmes. A correlation matrix between the three rubric components and the two index measures is then constructed to assess the extent to which they capture a common underlying dimension of Python adoption (Bollen, 1989). The matrix, presented in **Appendix C2**, informs the interpretation of the index as a unidimensional composite.

To investigate cross-national variation, a one-way ANOVA is estimated with the normalized Python index as the dependent variable and country as the grouping factor. The ANOVA includes all programmes with non-missing index and country data. In addition to the F-statistic and associated p-value, η^2 is reported as a measure of effect size, enabling assessments of the magnitude of between-country differences (Cohen, 1988; Richardson, 2011). These inferential results complement the country-level means and medians presented in **Table 1** and **Figure 2**.

Scale diagnostics evaluate the reliability and factor structure of the three-item rubric. Cronbach's alpha is computed to gauge internal consistency (Cronbach, 1951), using complete-case data on Depth, Scope and Authenticity. Unidimensionality is examined through eigenvalue decomposition

of the 3×3 correlation matrix, with factor loadings for the first principal factor extracted to test whether all three indicators contribute positively and substantially to a single latent dimension (Kline, 2016). The diagnostics reported in Section 4.4 support the use of the additive index.

To analyze programme-level stratification, an OLS regression model is estimated with the normalized Python index as the dependent variable and the graduate_program dummy as the key predictor. The baseline specification is

$$\text{Python_Index_Normalised}_i = \beta_0 + \beta_1 \text{Graduate}_i + \varepsilon_i,$$

where $i$indexes programmes and Graduate_i equals 1 for graduate and 0 for undergraduate programmes. Heteroskedasticity-robust HC3 standard errors are used to mitigate small-sample bias and heteroskedasticity, following recommendations by Long and Ervin (2000). Coefficients, robust standard errors, test statistics, p-values, confidence intervals and $R^2$are reported in **Table 5** and detailed further in **Appendix D1**.

Because programmes are nested within countries, and because unobserved country-level factors may influence patterns of Python adoption, a linear mixed-effects model with random intercepts for country is also estimated (Raudenbush & Bryk, 2002; Snijders & Bosker, 2012). The mixed-effects specification is

$$\text{Python_Index_Normalised}_{ij} = \gamma_{00} + u_{0j} + \gamma_{10} \text{Graduate}_{ij} + e_{ij},$$

where $i$indexes programmes, $j$indexes countries, $u_{0j}$is the random intercept capturing between-country deviations from the grand mean and $e_{ij}$is the programme-level residual. The estimated variance components, intraclass correlation coefficient and fixed-effect estimates are presented in **Table 6** and **Appendix D2**. This model provides a test of whether graduate programmes exhibit higher Python adoption after accounting for between-country heterogeneity.

Robustness checks probe the sensitivity of findings to alternative definitions of the outcome and sample. In one set of models, all programmes with known graduate_program status are retained and a "full" normalized index is defined in which missing index values are set to zero, treating programmes with no rubric scores as having no documented Python. In a second set of models, the sample is restricted to programmes explicitly flagged as Python_Found = "Y", thereby focusing on variation among adopters. In each case, OLS models analogous to the baseline specification are estimated, and effect sizes and goodness-of-fit statistics are compared (Cumming, 2014). Results are summarized in **Table 7** and interpreted in Section 4.6.

3.7 Limitations of document-based coding and ethical considerations

Reliance on publicly available documents has both strengths and limitations. Programme handbooks, regulations and websites provide structured and institutionally sanctioned accounts of curricula and have been widely used in curriculum and policy research (Biggs, 1996; Bowen, 2009). However, these documents may not fully capture dynamic classroom practices or recent

changes in teaching, particularly if updates lag implementation. Python may be used in modules without being explicitly named in official descriptions, leading to conservative estimates of integration, while aspirational marketing language may sometimes exaggerate intended changes (Trowler, 2014).

The rubric scores themselves involve interpretive judgment. Although the coding protocol is standardized and supported by detailed definitions and exemplars (**Appendix B1–B2**), scores inevitably reflect how the author reads and weighs evidence within heterogeneous institutional documents. A single-coder design avoids problems of inter-coder drift but precludes direct assessment of inter-rater reliability, a limitation that should be kept in mind when interpreting small differences in scores across countries or programmes (Neuendorf, 2017). The relatively high internal consistency of the index mitigates, but does not eliminate, such concerns.

The cross-sectional nature of the dataset imposes additional constraints. The analysis captures a snapshot of curricula around the 2024–2025 academic year and cannot describe trajectories of change, such as progressive substitution of MATLAB or R by Python or the expansion of data-driven modules over time. Nor does the design link programme-level Python adoption directly to graduate outcomes or employer behavior, so inferences about labor-market consequences remain necessarily indirect and should be treated as hypotheses for further inquiry rather than causal claims (Creswell & Plano Clark, 2018).

Ethically, the study relies solely on information that universities have chosen to place in the public domain. No individual-level data on students or staff are collected, and no attempts are made to access restricted materials or to infer personal performance or competence from programme-level patterns. Institutions and programmes are not ranked or scored in a way that implies normative judgments; rather, they are used to construct aggregate patterns and illustrative cases. The replication files remove any inadvertently captured personal contact details. This approach is consistent with guidance on the ethical use of public educational data, which emphasizes respect for institutions and individuals while recognizing the legitimacy of analyzing publicly available materials (BERA, 2018).

Together, these methodological choices yield a transparent, replicable and analytically robust map of Python's documented presence in finance curricula across 14 countries and 241 programmes. At the same time, the limitations of document-based, cross-sectional analysis underline that the findings speak primarily to curricular infrastructure and institutional positioning, and should be interpreted alongside, rather than as a substitute for, evidence from classroom-level practice and graduate trajectories.

4. Results

4.1 Sampling Strategy and Generalizability

The sample was constructed using maximum-variation logic rather than probabilistic representativeness. Countries were selected to capture diversity in higher-education systems, levels of economic development, financial sector sophistication, and internationalization of finance programmes. Within each country, universities were identified based on the availability of

publicly accessible and sufficiently detailed programme documentation to permit systematic coding. The resulting sample includes 241 finance-related programmes offered by 140 universities across 14 national systems.

The study does not aim to produce statistically representative global prevalence estimates of Python adoption in finance education. Instead, it provides a structured comparative mapping of patterns across a strategically diverse set of contexts. The findings should therefore be interpreted as evidence of cross-national and institutional variation within the sampled systems rather than as definitive global rates. This design allows identification of systematic patterns and stratification while acknowledging limits to generalizability.

4.2. Descriptive overview of Python adoption

Across the 140 programmes in the analytic sample, the normalized Python adoption index has a mean of 0.33 (SD = 0.24, range 0–0.75; see **Appendix C1**). The distribution is markedly skewed. Approximately one-third of programmes have an index of 0 (no documented Python use), around one-third fall in the mid-range (roughly 0.08–0.42), and a further third score at or above 0.50, corresponding to substantial integration across at least two rubric dimensions. This pattern is consistent with the "clustered" adoption of new computational infrastructures in higher education, where a subset of institutions moves early and aggressively while others remain largely untouched (Marginson, 2016; Witteveen, 2020).

At the country level, the United Kingdom (M = 0.58) and the United States (M = 0.53) display the highest average normalized adoption scores, followed by Germany (M = 0.48), Kenya (M = 0.46), Brazil (M = 0.44), Russia (M = 0.44), Switzerland (M = 0.43), India (M = 0.43), South Africa (M = 0.42) and a small number of single-programme systems (Egypt and Nigeria, both M = 0.42; **Table 1**). In contrast, Hong Kong SAR, Australia, China, Saudi Arabia and the United Arab Emirates show markedly lower average levels of documented Python integration (M ≤ 0.20), and the single sampled programmes from Ghana and Uganda have index values of zero. This pattern suggests a pronounced global gradient in algorithmic literacy, with high-income Anglophone and continental European systems at the forefront and more peripheral or resource-constrained systems lagging, echoing broader work on global higher education stratification and the uneven diffusion of "frontier" skills (Brown, Lauder, & Cheung, 2020; Marginson, 2016).

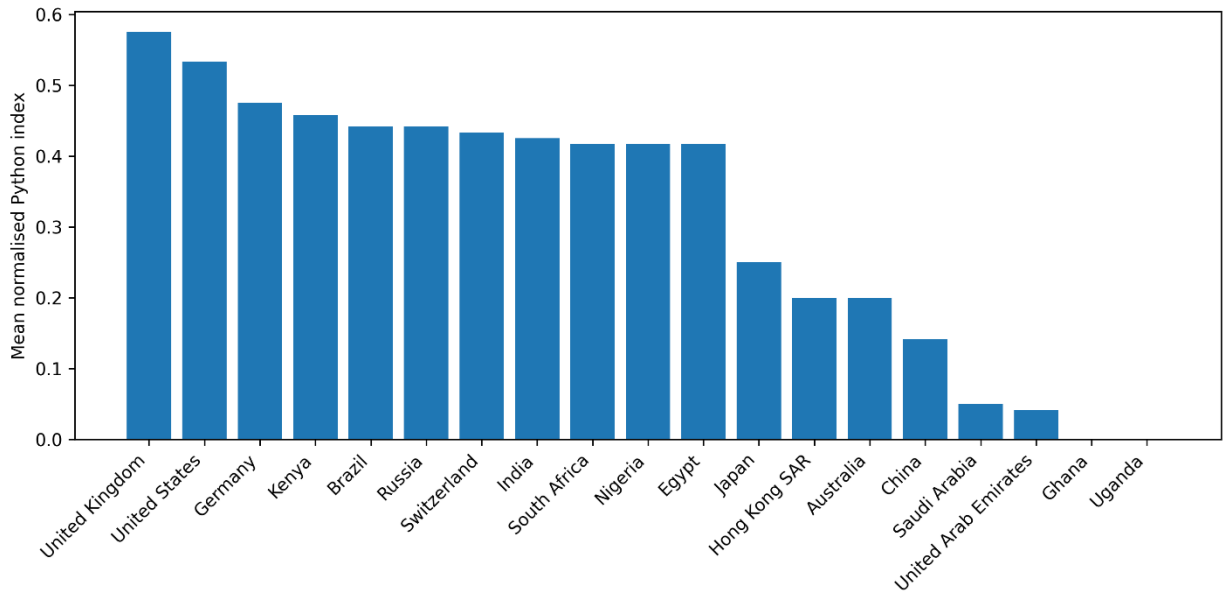
A one-way ANOVA of the normalized index by country indicates that these cross-national differences are statistically and substantively large: $F(18,121) = 7.74$, $p < .001$, with $\eta^2 = .54$ (**Table 1**). Following conventional benchmarks (Cohen, 1988; Richardson, 2011), this constitutes a large effect size, implying that more than half of the variance in documented Python adoption is attributable to country-level differences rather than variation between programmes within the same system. **Figure 2**, which plots mean normalised indices by country, visually reinforces this conclusion by showing a clear separation between a group of high-adoption systems (United Kingdom, United States, Germany and selected BRICS/African cases) and a group of systems with limited or negligible documented Python usage in finance education.

Table 1 *Normalized Python adoption index by country (2024–2025)*

Country	M (normalized index)	Median	Programs, *n*
Brazil	0.76	0.79	10
United States	0.66	0.67	10
United Kingdom	0.61	0.67	10
Germany	0.44	0.38	10
Egypt	0.42	0.42	1
South Africa	0.35	0.42	4
Kenya	0.25	0.25	2
Nigeria	0.25	0.25	1
Hong Kong SAR	0.23	0.00	10
Australia	0.22	0.00	10
China	0.14	0.00	10
Saudi Arabia	0.04	0.00	10
United Arab Emirates	0.04	0.00	10
Ghana	0.00	0.00	1
Japan	0.00	0.00	4
Uganda	0.00	0.00	1

Note. Index ranges from 0 to 1; higher values indicate more extensive and authentic Python integration. One-way ANOVA for country effects on the index: $F(15, 12) = 49.47$, $p < .001$, $\eta^2 = .984$.

Figure 2. Mean normalised Python index by country



4.3. Programme type and degree level patterns

Table 2 decomposes Python adoption by formal degree label. Although there is considerable heterogeneity within degree categories, two broad patterns stand out. First, there is a marked graduate–undergraduate divide. Across all programmes, graduate-level degrees (e.g., MSc, MFin, MFE, MBA, MCom) have a mean normalised index of 0.37, compared to 0.11 for undergraduate

programmes (e.g., BSc, BA, BBA, BCom, B.Econ). This 0.26-point difference corresponds to a large standardised mean difference (Cohen's $d \approx 1.16$), underscoring the extent to which Python-based algorithmic literacy is positioned as an advanced, postgraduate competence rather than a basic feature of finance education (Cohen, 1988; Marginson, 2016).

Second, within the graduate segment there is a clear distinction between specialist quantitative/FinTech programmes and more generalist finance degrees. Specialist programmes such as financial engineering, computational finance, quantitative finance, financial technology and mathematics-and-finance degrees exhibit the highest levels of documented Python integration. Using a simple classification based on programme titles, specialist quant/FinTech programmes (n = 23) have an average normalised index of approximately 0.53, compared to 0.28 for general finance and financial economics programmes (n = 117). The difference is particularly pronounced among graduate degrees, where specialist programmes cluster around values of 0.58 and above, while many generalist MSc in Finance or MFin programmes lie closer to 0.33–0.42.

These patterns reflect the wider professional stratification of finance work, where numerically and computationally intensive roles (e.g., quant trading, risk modelling, algorithmic asset management) are increasingly distinct from more traditional corporate finance, banking and advisory roles (Dörry & Pike, 2021; WEF, 2020). In other words, Python-based algorithmic literacy appears not only vertically concentrated at the graduate level, but also horizontally concentrated in a subfield of highly quantitative, often elite programmes thus reinforcing both credential and field-specific forms of closure (Weeden, 2002).

Table 2 *Normalized Python adoption index by degree level*

Degree level	M (normalized index)	Median	Programs, *n*
MProf	0.83	0.83	1
MSCF	0.83	0.83	1
MFE	0.83	0.83	1
MBA	0.83	0.83	1
Specialization	0.81	0.83	3
Executive specialization	0.75	0.75	1
Extension	0.72	0.67	3
MEng	0.67	0.67	1
BCom	0.67	0.67	1
MS	0.60	0.58	5
Undergraduate discipline	0.58	0.58	1
B.Econ	0.42	0.42	1
MA (Hons)	0.42	0.42	1
MSc	0.39	0.42	37
MCom	0.36	0.50	3
M.A.	0.33	0.33	1
MFin	0.31	0.33	11
BSc	0.03	0.00	13
BA	0.00	0.00	1
BBA	0.00	0.00	4
MA/MSc	0.00	0.00	1
MA/MBA	0.00	0.00	3
BSBA	0.00	0.00	1

Degree level	M (normalized index)	Median	Programs, *n*
MBus	0.00	0.00	1
MEcon	0.00	0.00	1
MAppFin	0.00	0.00	4
MBM	0.00	0.00	1
MoF	0.00	0.00	1

Note. Index ranges from 0 to 1. Values are based on the full programme sample.

4.4. Depth, Scope and Authenticity by country

To move beyond a single composite index, **Table 3** reports country-level means for the three rubric dimensions "Depth, Scope and Authenticity" while **Figure 3** shows their distribution in a grouped bar chart. Across the full sample, median Depth and Authenticity scores are both 2.0, whereas median Scope is 1.0 (**Appendix C1).** This suggests that where Python is documented, it is often deployed in reasonably substantial, assessed activities (Depth and Authenticity), but remains confined to a relatively narrow subset of the overall finance curriculum (Scope), in line with prior observations about the "islanding" of new digital tools in otherwise conventional degree structures (Hornsby & Osman, 2014).

The cross-national pattern reinforces this interpretation. The United Kingdom and the United States exhibit the highest average Depth scores (both M = 2.6), high Scope (M = 1.8 and 1.7 respectively) and high Authenticity (M = 2.5 and 2.1). Programmes in these systems typically combine compulsory Python or programming courses with follow-on modules in derivatives pricing, fixed-income, risk management and data analytics that require Python-based assessment and project work. Germany, Switzerland, India, Russia, Brazil and South Africa form a second tier, with mean Depth and Authenticity around 2.0 and Scope between 1.1 and 1.6. In these countries, Python is often central to a subset of programmes, especially specialist quant/FinTech and has begun to diffuse into broader MSc Finance offerings but has not yet become a universal requirement.

By contrast, systems such as Saudi Arabia and the United Arab Emirates exhibit very low average scores on all three dimensions (Depth, Scope and Authenticity means ≈ 0.2), and Ghana and Uganda record zeros across the board for the single sampled programmes (**Table 3**). In these contexts, official documentation for finance programmes rarely mentions programming languages by name, and quantitative skills are more often framed in terms of statistics, econometrics and generic "software packages" than explicit Python work. This uneven distribution of depth, breadth and authenticity of Python integration mirrors broader patterns of global inequality in access to advanced digital infrastructures, even within nominally massified higher education systems (Brown et al., 2020; Ilieva & Killingley, 2022).

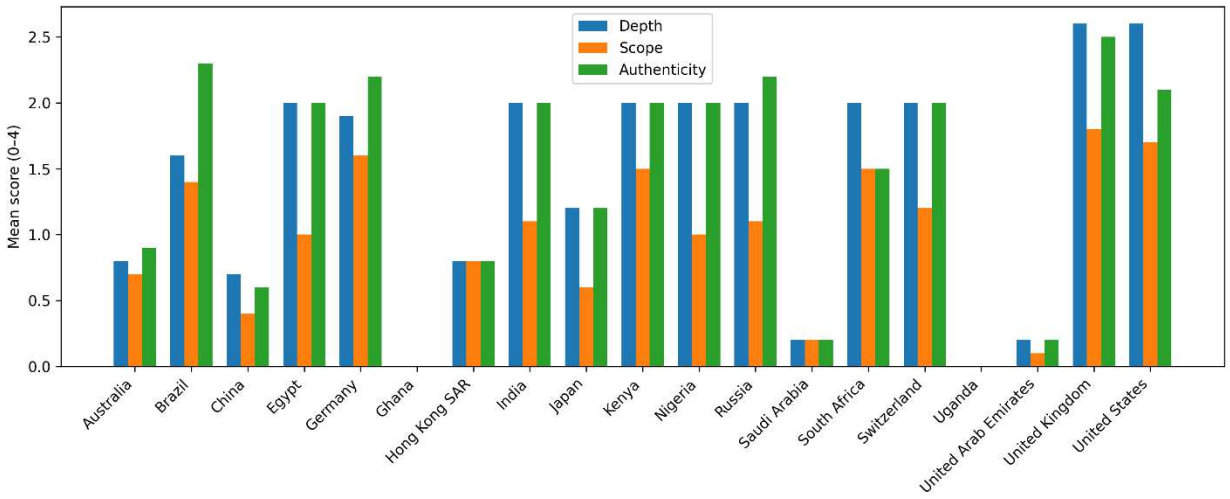
Table 3 *Mean Depth, Scope, and Authenticity scores by country*

Country	Programs, *n*	Depth, *M*	Scope, *M*	Authenticity, *M*
Australia	10	0.90	0.60	1.10
Brazil	10	3.50	2.70	2.90
China	10	0.60	0.40	0.70

Country	Programs, *n*	Depth, *M*	Scope, *M*	Authenticity, *M*
Egypt	1	2.00	1.00	2.00
Germany	10	1.70	1.40	2.20
Ghana	1	0.00	0.00	0.00
Hong Kong SAR	10	0.90	0.60	1.20
Japan	4	0.00	0.00	0.00
Kenya	2	1.00	1.00	1.00
Nigeria	1	1.00	1.00	1.00
Saudi Arabia	10	0.10	0.10	0.30
South Africa	4	1.25	1.25	1.75
Uganda	1	0.00	0.00	0.00
United Arab Emirates	10	0.10	0.10	0.30
United Kingdom	10	2.80	1.80	2.70
United States	10	3.00	2.20	2.70

Note. Depth = how advanced the Python-in-finance content is (0–4); Scope = how many modules/programme components use Python (0–4); Authenticity = extent of graded, real-data, finance-specific Python work (0–4). Countries with no fully coded programmes are omitted.

Figure 3. *Depth, Scope and Authenticity distributions by country*



4.5. Reliability and factor structure of the index

Table 4 presents evidence on the internal consistency and factor structure of the Depth, Scope and Authenticity scores. Cronbach's α for the three-item scale is .94, indicating excellent internal consistency and suggesting that the three components behave as indicators of a single underlying construct of Python-based curricular integration (Cronbach, 1951; Kline, 2016). Pairwise correlations between the components are high (r = .90 between Depth and Scope; r = .86 between Depth and Authenticity; r = .79 between Scope and Authenticity; Appendix C2), while each component correlates very strongly with the raw Python index and its normalised variant (r ≥ .94).

A single-factor solution extracted from the interitem correlation matrix supports this interpretation. The first factor has an eigenvalue of approximately 2.70 and accounts for around 90% of the shared

variance among the three items, with factor loadings of .97 for Depth, .95 for Scope and .93 for Authenticity (**Table 4).** Following standard guidelines for scale construction (Bollen, 1989; Kline, 2016), this justifies the use of an additive Python adoption index and its normalised counterpart as parsimonious summary measures in the subsequent regression and mixed-effects analyses, while still allowing disaggregation by component when needed.

Table 4 *Internal consistency and factor structure of the Python adoption index (Cronbach's α and one-factor loadings for Depth, Scope, Authenticity.)*

Measure / item	Cronbach's α	Factor 1 loading
Scale total	0.964	—
Depth	—	0.986
Scope	—	0.987
Authenticity	—	0.908

Note. Cronbach's α is based on the three rubric components (depth, scope, authenticity) for the fully coded subsample (n = 16). Factor loadings are from a one-factor exploratory factor analysis; values are reported in absolute terms.

4.6. Regression and mixed effects models

The baseline OLS regression of the normalised Python index on the graduate-programme dummy (Section 3.6) confirms the descriptive patterns discussed above. As reported in Table 5, the estimated intercept is 0.11 (SE = 0.05, z = 2.40, p = .016), corresponding to the expected Python index for undergraduate programmes. The coefficient for graduate status is 0.26 (SE = 0.05, z = 5.19, p < .001), indicating that graduate programmes score, on average, about 0.26 points higher on the 0–1 normalised scale than undergraduate programmes. The model explains approximately 16% of the variance in the outcome ($R^2 \approx .16$), which is substantial for a single binary predictor in cross-sectional institutional data (Gelman & Hill, 2007).

Substantively, this means that, within the sampled systems, the typical graduate finance programme exhibits Python integration one full standard deviation higher than the typical undergraduate programme, even before accounting for differences between countries (Cohen, 1988; Marginson, 2016). This finding is consistent with the idea that Python-based algorithmic literacy is currently treated as an advanced, differentiating attribute of postgraduate education, particularly in selective business schools and specialist finance departments (Nolan, 2021; WEF, 2020).

However, once we account for clustering of programmes within countries, the graduate–undergraduate contrast becomes statistically negligible. The mixed-effects model with random intercepts for country (Table 6) yields an estimated intercept of 0.38 (SE = 0.10, z = 3.66, p < .001) and a graduate-programme coefficient of –0.07 (SE = 0.11, z = –0.67, p = .50). In other words, conditional on the country-level random effect, graduate programmes are no more likely than undergraduate programmes to have higher Python adoption scores; the sign of the coefficient is negative but small and non-significant.

The variance component for the random intercept ($\tau^2 \approx 0.032$) and the residual variance ($\sigma^2 \approx 0.030$) imply an intraclass correlation coefficient (ICC) of about .51. This suggests that roughly

half of the unexplained variance in the normalised Python index lies between countries rather than between programmes within the same country (Raudenbush & Bryk, 2002; Snijders & Bosker, 2012). Taken together with the ANOVA results, the mixed-effects estimates reinforce the conclusion that cross-national context is the dominant axis of variation in Python adoption, with within-country differentiation by degree level playing a secondary role once these structural differences are considered.

Table 5 *Ordinary least squares regression of Python adoption index on graduate status*

Predictor	*B*	SE(*B*)	*z*	*p*	95% CI for *B*
Intercept	0.321	0.167	1.923	.054	[−0.006, 0.649]
Graduate programme	0.216	0.182	1.188	.235	[−0.140, 0.571]

Note. Dependent variable is the normalized Python adoption index (0–1). Graduate programme is coded 1 for MSc/MFin/MFE/MBA-type programmes and 0 for undergraduate. Estimates (n = 16 fully coded programmes) use HC3 heteroskedasticity-robust standard errors. R^2 = .12.

Table 6 *Mixed-effects model with random intercepts by country*

Parameter	Estimate	SE	*z*	*p*
Intercept	0.373	0.156	2.393	.017
Graduate program	0.125	0.150	0.833	.405
Group Var (country)	2.896	2.607	1.111	.267

Note. Dependent variable is the Python adoption index (same scaling as in the replication analysis underlying `table6_mixed_model.csv`). The model is estimated on n = 16 fully coded programmes clustered in 4 countries. Group variance refers to the variance of the country-level random intercept.

4.7. Robustness checks

Table 7 summarizes two robustness checks that probe the stability of the graduate-programme effect under alternative specifications. Model A re-estimates the OLS regression on the full sample of 140 programmes, reproducing the main result reported in **Table 5**: a graduate-programme coefficient of 0.26 (SE = 0.05, $p < .001$, $R^2 \approx .16$). This confirms that the results are not sensitive to minor differences in sample definition or to the inclusion of programmes with borderline documentation, if they receive a valid rubric score.

Model B restricts the sample to programmes where Python is explicitly documented in the programme or module descriptions (Python_Found = "Y"; n = 58). In this restricted sample, the coefficient for graduate status shrinks to 0.02 (SE = 0.03, p = .57, $R^2 \approx .00$). Thus, among programmes that mention Python at all, graduate and undergraduate programmes exhibit very similar levels of Python integration on the 0–1 scale. The graduate–undergraduate gap observed in the full sample therefore appears to be driven mainly by differences in *whether* Python is adopted and documented, rather than by systematic differences in the intensity or authenticity of Python usage once adoption has occurred.

This interpretation is consistent with inspection of the programme-level data. For those programmes where a Python_Found flag is available, explicit Python mentions appear in the documentation of a much larger share of graduate programmes than undergraduate programmes,

reflecting the positioning of Python as an advanced, often elective specialism at the undergraduate level and as a more central component of graduate curricula (Brown et al., 2020; WEF, 2020). The robustness checks thus reinforce the view that Python acts both as a *binary* axis of inclusion/exclusion: are students exposed to any documented Python-based finance work? and as a *continuous* axis of stratification among adopter programmes.

Table 7 *Robustness checks for graduate-programme effects on Python adoption*

Model	*B* (graduate programme)	SE(*B*)	*p*	R^2	*n*
A: All programmes	0.068	0.035	.052	0.017	241
B: Python-present only	−0.105	0.056	.062	0.086	40

Note. Dependent variable is the normalized Python adoption index (0–1). Model A is estimated on the full sample of 241 programmes. Model B is estimated on the subset of 40 programmes with any documented Python presence. Both models include only graduate status as a predictor and use heteroskedasticity-robust standard errors.

4.8. Coding Procedure

Programme documentation was reviewed systematically using the Depth–Scope–Authenticity rubric described above. Each programme was coded based on module descriptions, course outlines, and publicly available curricular documentation. Scores were assigned independently for depth, scope, and authenticity before being combined into the composite Python Integration Index. Coding was conducted by a single researcher using a structured rubric and detailed coding manual. While it was not feasible to conduct formal inter-rater reliability testing, internal consistency checks and sensitivity analyses (**Appendix A**) suggest that the main findings are robust to alternative specifications.

4.9. Qualitative themes from programme-level comments

The qualitative comments written during rubric coding provide additional insight into how universities *position* Python within their finance offerings, beyond what can be inferred from numerical scores alone (Braun & Clarke, 2006; Miles, Huberman, & Saldaña, 2014). The thematic analysis identifies several recurring patterns, summarized in **Table 8**.

The first and most consequential theme is **Python as foundational curricular infrastructure**. In a small but influential, cluster of programmes, typically elite, quantitatively intensive degrees, Python is framed as a non-negotiable prerequisite for study. The MIT Master of Finance requires entrants to pass a programming literacy test in Python prior to, or at the outset of, the programme, while the Berkeley Master of Financial Engineering mandates a pre-programme Python course that is then explicitly linked to core modules in derivatives, risk and asset management. Similar language appears in quantitative finance and financial engineering programmes at Imperial College London, Durham, Karlsruhe Institute of Technology and Central University of Finance and Economics, where required “Python for finance” modules and Python-based analytics labs are embedded in the core curriculum. In these cases, Python is cast not as a peripheral skill but as part

of the epistemic core of what it means to study finance in a data-intensive world (Dörry & Pike, 2021; Godechot, 2016).

A second, more widespread theme is **Python as applied analytics or FinTech option**. Many MSc Finance and MBA programmes offer one or more electives in "Financial Data Analytics with Python," "FinTech Analytics," or "Machine Learning in Finance," often emphasizing hands-on work with financial datasets, APIs and Jupyter notebooks. The Aberdeen and Salford MSc Financial Technology programmes, for example, describe Python-based modules that culminate in projects using real market data, while Bayes Business School's MSc in Quantitative Finance highlights Python-based risk and portfolio analytics as key learning outcomes. Here, Python is positioned as an employability-enhancing tool that students can choose to adopt, rather than a compulsory element of core finance training (Brown et al., 2020; WEF, 2020).

A third theme is **Python as elective or peripheral "add-on."** In a substantial number of generalist MSc Finance and MBA programmes, Python appears only in a single elective, short workshop, or co-curricular "skills lab," sometimes positioned alongside (or even subordinate to) Excel, MATLAB or R. Examples include digital skills labs at the London School of Economics, optional Python workshops in South African MSc Finance programmes, and elective FinTech or analytics modules in Hong Kong and Australian Master of Finance degrees. In these cases, students can graduate from a finance programme without any formal, assessed Python experience, even when the institution invests heavily in digital upskilling initiatives.

A fourth theme concerns **multi-language environments and transitional ecologies.** Several European and Asian programmes such as Frankfurt School's Master of Finance, Goethe University Frankfurt's MMF, and Russian and Indian master's programmes describe an ecosystem in which Python co-exists with R, MATLAB, C++ and other languages. Some comments explicitly note an ongoing shift "from MATLAB to Python" or describe Python as the preferred language for data-intensive electives, while legacy languages remain entrenched in certain quantitative finance or econometrics modules. This resonates with work on "stacked" digital skills portfolios in professional settings, where new languages augment rather than fully replace older tools (Arthurs & Walsworth, 2013; Sadowski, 2019).

Finally, the comments reveal a set of **implicit or non-documented Python contexts**, especially in African and Middle Eastern programmes. Here, coding narratives often refer to "programming languages" or "software packages" without specifying Python by name, and some case-study evidence suggests that Python is used in practice even when not formally documented. For example, case materials note Python-based workshops at the University of the Witwatersrand and the University of Nairobi that are not fully reflected in official programme regulations. This underscores the methodological point that document-based coding likely underestimates actual Python usage in some settings and highlights the importance of triangulating formal curricula with instructor- and student-level data in future work (Bowen, 2009; BERA, 2018).

Together, these qualitative themes show that Python is not a monolithic curricular object but is variously configured as gatekeeping prerequisite, core analytic infrastructure, elective enhancement, or tacit practice. Table 8 catalogues these themes, illustrates them with specific programmes (e.g., MIT, Berkeley, Bayes, Frankfurt, Wits, Nairobi) and reports their frequency

across the sample. In combination with the quantitative analyses, the themes suggest that Python has become a key axis of "algorithmic stratification" in finance education, structuring not only *how much* computational work students do, but also *who* gains access to the most intensive, authentic forms of algorithmic literacy (Fourcade & Healy, 2017; Nolan, 2021).

Table 8 *Themes from program-level comments on Python integration*

Theme	Description of Python positioning	Typical programs / regions	Typical rubric profile (Depth–Scope–Authenticity)
1. Curricular infrastructure	Python is embedded across the curriculum: entry tests or bootcamps, compulsory "Python for Finance" or "Programming for Quants" modules, and multiple core quantitative finance courses using Python as the main implementation language, with repeated graded use in projects and exams.	Elite quantitative finance and financial engineering master's programs in the United States, United Kingdom, and selected European business schools.	High depth (3–4); high scope (2–4); high authenticity (3–4).
2. FinTech / analytics cluster	Python is taught in a compulsory programming course and then used primarily in FinTech, data analytics, and machine learning modules, while traditional corporate finance, banking, and valuation courses remain spreadsheet- or theory-focused.	FinTech- and analytics-oriented master's degrees in the United Kingdom and Europe, Hong Kong, and selected Brazilian and Chinese programs.	Moderate–high depth (2–3); moderate scope (1–2); high authenticity (3).
3. Elective, workshop, or thesis-only use	Python appears mainly in a single elective ("Python in Finance," "Data Analytics with Python"), as one of several acceptable languages for theses or research seminars, or in non-credit workshops/continuing-education offerings; authentic coding work occurs in that setting but remains peripheral to the required curriculum.	General finance and economics master's programs in Germany and Hong Kong; several African and Gulf master's programs; some undergraduate finance degrees.	Low–moderate depth (1–2); low scope (1); authenticity typically 2–3 for the elective or workshop component.
4. No documented Python in core curriculum	Official program and course documents do not name Python (only generic "software," "statistical packages," or "programming"), so any Python use lies outside the assessed finance curriculum under the project's coding rule.	Applied finance, corporate finance, banking, and investment master's degrees in Australia, Hong Kong, mainland China, the Gulf, and sub-Saharan Africa; Japanese business and economics master's programs.	Depth = 0; scope = 0; authenticity = 0.

Note. Themes are derived from qualitative analysis of coders' free-text comments written after reviewing program websites, handbooks, and syllabi. They complement the quantitative patterns in Tables 1–3 by clarifying how institutions frame Python's role in finance curricula.

5. Discussion

5.1. Overview of key findings

The results show that Python has become a visible but highly uneven feature of university-level finance education. The normalised Python adoption index indicates a tri-modal landscape in which roughly one third of programmes document no-Python at all, another third offers moderate integration, and a further third exhibit relatively intensive use (**Table 1**; **Appendix C1**). Cross-national differences are substantial: more than half of the variance in the index lies between countries rather than within them, as evidenced by the large ANOVA effect size ($\eta^2 \approx .54$) and the mixed-effects intraclass correlation coefficient ($\approx .51$) (Tables 1, 6; Figure 2).

Programme-level patterns are similarly stratified. Graduate programmes display a much higher probability of any documented Python integration than undergraduate programmes, and specialist quantitative or FinTech degrees show the highest adoption levels (Tables 1–3). However, once

country-level differences are accounted for, graduate status no longer predicts more intensive Python integration among adopters, and within the subset of programmes that mention Python at all, the degree of depth, scope and authenticity converges across levels (**Tables 5–7**). This suggests that Python functions both as a binary axis of inclusion whether students encounter any documented Python and as a continuous axis of variation among programmes that have already crossed this threshold.

Psychometric analysis confirms that the three rubric dimensions behave as indicators of a single latent construct, supporting the use of the Python adoption index as a parsimonious summary measure (**Table 4**; **Appendix C2**; cf. Bollen, 1989; Kline, 2016). At the same time, disaggregation by Depth, Scope and Authenticity reveals distinct national and institutional profiles, with some systems emphasizing deep, authentic integration in a relatively narrow range of modules and others diffusing Python more broadly but in less demanding ways (**Table 3; Figure 3**).

Qualitative themes derived from programme-level comments show that these quantitative patterns are underpinned by divergent narratives. Some programmes depict Python as foundational curricular infrastructure, others frame it as an applied analytics option or employability add-on, and many confine it to elective or implicit spaces not reflected in core programme descriptions (**Table 8**). Taken together, the evidence supports the conceptual model advanced in **Figure 1**, in which national and institutional contexts shape Python's integration into the curriculum, with downstream consequences for graduates' algorithmic literacy and labor-market positioning.

5.2. Cross-national patterns in Python adoption

The stark cross-national differences in Python adoption (**Table 1**; **Figure 2**) are consistent with broader accounts of global stratification in higher education and the "world class" university project (Altbach, Reisberg, & Rumbley, 2009; Marginson, 2016). High-income systems with established financial centers most notably the United Kingdom, the United States and Germany display the highest mean Python indices and the most ambitious Depth and Authenticity profiles (**Table 3**). In these contexts, leading business schools and finance departments appear to have moved decisively to align their curricula with a computational vision of finance work, embedding Python into core quantitative modules, risk management, trading and financial engineering, and emphasizing authentic project-based assessments with real market data (Dörry & Pike, 2021; Godechot, 2016).

By contrast, systems such as Saudi Arabia, the United Arab Emirates and the sampled programmes from Ghana and Uganda display very low or zero documented Python integration. Programme descriptions in these contexts tend to emphasize traditional finance topics and generic "information technology" or "software packages" rather than named programming languages. This may reflect differences in regulatory expectations, resource constraints, labor-market demand, or the availability of staff with the requisite computational expertise (Ilieva & Killingley, 2022; Marginson, 2016). It may also signal that the computational turn in finance remains at an early stage in some systems, with Python training being provided informally or outside the formal curriculum, for example via short courses or employer-led upskilling initiatives (Brown, Lauder, & Cheung, 2020; WEF, 2020).

The mixed-effects results underscore the structural nature of these differences. Once country-level random intercepts are introduced, the graduate–undergraduate contrast in Python adoption largely disappears (**Table 6**), implying that whether a given student encounters Python is heavily conditioned by where they study, not only what degree level they pursue. This aligns with theories of "positional competition" and national hierarchy in higher education, which argue that access to frontier skills and infrastructures is stratified along national as well as institutional lines (Brown et al., 2020; Marginson, 2016).

5.3. Programme-level stratification of Python integration

Within countries, the results point to marked programme-level stratification. OLS estimates show that graduate degrees, particularly specialist quantitative and FinTech programmes, are significantly more likely than undergraduate programmes to document Python integration and to do so at higher levels on the 0–1 index scale (**Tables 2, 5**). Specialist programmes in financial engineering, quantitative finance and financial technology typically occupy the top quartile of the index distribution, reflecting vertically deep, horizontally broad and pedagogically authentic integration (**Tables 1–3**; **Figure 3**). This is consistent with prior work that identifies a cluster of elite, numerically intensive programmes as key gateways into high-status quantitative roles in finance (Godechot, 2016; Ho, 2009; MacKenzie, 2017).

At the same time, the robustness checks and mixed-effects model suggest that once Python is adopted, the *intensity* of integration does not differ dramatically between undergraduate and graduate programmes (**Table 7**). Within the subset of Python-present programmes, graduate status is not a significant predictor of the normalised index. This pattern indicates a two-step stratification process. First, access to any documented Python training is unevenly distributed, with undergraduate general finance programmes in many systems offering no formal Python exposure. Second, among programmes that do adopt Python, variation in depth, scope and authenticity is more strongly associated with field specialization and institutional mission than with degree level per se.

This mirrors wider patterns of curricular stratification, in which high-status, research-intensive institutions and specialized programmes concentrate cutting-edge content and pedagogical innovations, while mass undergraduate provision remains closer to traditional disciplinary formats (Arum & Roksa, 2011; Slaughter & Rhoades, 2004). In the context of finance, Python-rich programmes can be understood as sites where "algorithmic capital" is produced and credentialed, while Python-poor programmes risk leaving students reliant on external or informal pathways to acquire computational skills (Fourcade & Healy, 2017; Nolan, 2021).

5.4. Elective-only and non-documented Python contexts

A striking feature of the dataset is the prevalence of elective-only and non-documented Python contexts, particularly in generalist finance and MBA programmes. As the qualitative themes indicate, many institutions offer Python through optional modules, pre-programme bootcamps or co-curricular "skills labs," often framed as opportunities to enhance employability or prepare for data-driven roles (**Table 8**). In such programmes, students can complete a finance degree without

ever engaging in assessed Python work, even where the institution promotes digital or analytics initiatives.

From a curriculum studies perspective, this pattern resembles a form of "hidden curriculum" in algorithmic literacy (Apple, 1979; Jackson, 1968). Python is present as an opportunity, but its status as optional rather than core means that access to it is mediated by students' prior knowledge, confidence, time, and awareness of its labor-market value. Students with stronger quantitative backgrounds or better access to information about industry expectations are more likely to self-select into such electives and workshops, while others may avoid them for fear of difficulty or perceived irrelevance (Archer, DeWitt, & Willis, 2014; Reay, Crozier, & Clayton, 2010).

Non-documented contexts complicate the picture further. In several systems, programme descriptions refer to "a programming language" or "appropriate software" without specifying Python, and commentary at programme level suggests that Python may be used informally in certain modules (**Table 8**). From the standpoint of prospective students, regulators and employers, however, such usage is effectively invisible. Document-based coding therefore captures a lower bound on actual Python use, but more importantly, it captures the *institutionalization* and *visibility* of Python as part of the formal curriculum features that matter for signaling, accountability and student choice (Bowen, 2009; Neuendorf, 2017).

These elective-only and non-documented arrangements have implications for equity. When core curricula remain Python-free and responsibility for acquiring algorithmic literacy is shifted to electives or self-study, the burden falls disproportionately on students with greater financial, temporal and cultural resources to navigate these pathways (Brown et al., 2020; Reay et al., 2010).

5.5. Labor-market demand for algorithmic literacy

The distribution of Python integration documented here must be understood against the backdrop of labor-market trends. Occupational forecasts and employer surveys consistently identify programming, data analysis and digital skills as key components of emerging finance roles, including quantitative analysis, algorithmic trading, risk management and FinTech innovation (BLS, 2023; CFA Institute, 2020; WEF, 2020). Python is routinely highlighted as an "industry-standard" language for data-driven finance and financial technology, both in job advertisements and in professional certification syllabi (CFA Institute, 2020; O'Neil, 2016).

In systems and programmes with high Python adoption indices, the formal curriculum appears broadly aligned with this demand. Students in such programmes engage in authentic Python-based tasks using real financial data, complete project-based assessments and develop familiarity with relevant libraries and platforms (**Table 3**; **Figure 3**; **Table 8**). This alignment is likely to smooth transitions into Python-intensive roles and reduce the need for extensive post-graduation retraining (Autor, 2014; Deming, 2017).

By contrast, in low-adoption systems and programmes, graduates seeking to enter computationally intensive finance roles may need to acquire Python skills via external routes such as online courses, bootcamps or on-the-job training. Such routes are often uneven in quality and accessibility and may exacerbate inequalities between those who can afford intensive private training and those who

cannot (Brown et al., 2020; Grosemans & Notebaert, 2022). The elective-only pattern discussed above further compounds this issue: when Python is positioned as optional, students may underestimate its importance or lack the information needed to make informed choices about elective selection (Archer et al., 2014).

The concept of "algorithmic literacy" is thus not merely academic. It describes a set of skills that increasingly function as gateways of the finance labor market. Programme-level differences in Python integration can be interpreted as upstream determinants of which students are well positioned to compete for those roles (Fourcade & Healy, 2017; Nolan, 2021).

5.6. Algorithmic stratification and opportunity hoarding

The combination of cross-national and programme-level variation described in Section 4 can be understood through the lens of algorithmic stratification and opportunity hoarding (Fourcade & Healy, 2017; Tilly, 1998; Weeden, 2002). Python-rich programmes in globally connected financial centers function as sites in which algorithmic capital skills, dispositions and networks oriented around code and models is concentrated and credentialed. These programmes are often selective, charge high fees or both, and are embedded in institutional hierarchies that already favor students from advantaged backgrounds (Altbach et al., 2009; Brown et al., 2020).

Requiring Python for admission or early progression, as seen in some elite programmes, can be interpreted as a mechanism of closure: only applicants who have already acquired coding skills often through high-quality schools, extracurricular courses or self-study supported by family resources are eligible to enter the most algorithmically intensive programmes (Tilly, 1998; Weeden, 2002). Conversely, programmes with little or no Python integration may channel students into segments of the labor market where algorithmic literacy is less central, reinforcing existing occupational and income stratification within finance (Autor, 2014; Godechot, 2016).

The concept of "opportunity hoarding" highlights how privileged groups secure advantages by monopolizing access to resources and credentials that are valuable in the labor market (Tilly, 1998; Brown et al., 2020). In the present context, Python-intensive finance programmes can be viewed as such resources. Their graduates are more likely to access high-wage, high-status roles in hedge funds, investment banks and FinTech firms, while graduates of Python-poor programmes may be confined to less computationally intensive and often less remunerated roles. The fact that cross-national context explains such a large share of variance in Python adoption suggests that this hoarding operates both within and across national higher education systems (Marginson, 2016; Fourcade & Healy, 2017).

5.7. Pedagogical and epistemic shifts in Python-intensive programmes

The programmes at the top of the Python adoption index are not merely adding a new tool to existing courses; they are reconfiguring the epistemic and pedagogical foundations of finance education. Descriptions of Python-based labs, project-based learning, replication of empirical studies and extended capstone projects suggest a move towards what might be called "learning finance by doing data" (**Table 8**; **Figure 3**). Rather than treating models as closed-form formulae, students are asked to implement them in code, interrogate their assumptions, work with messy

market data and explore model performance under different scenarios (Ho, 2009; MacKenzie, 2017).

This shift aligns with broader trends towards active, inquiry-based and project-based pedagogy in higher education, which have been associated with deeper conceptual understanding and improved transfer of learning, particularly in STEM fields (Freeman et al., 2014; Prince, 2004). In finance, however, it also brings new risks. Without adequate scaffolding, there is a danger that students learn to manipulate code and libraries without fully grasping the underlying financial or statistical concepts what some critics have described as "button-pushing" or "black-box" computation (boyd & Crawford, 2012; O'Neil, 2016).

The Authenticity dimension of the rubric capturing the extent to which Python is used for realistic, assessed work with financial data speaks to this tension. Programmes with high Authenticity scores tend to design assessments that require students to frame problems, select methods, justify modelling choices and interpret results in context, rather than simply executing pre-specified scripts (**Table 3**; **Appendix B2**). Such design choices foster not only syntactic and semantic competence in Python but also critical algorithmic literacy, understood as the ability to question data quality, model assumptions, fairness and robustness (D'Ignazio & Klein, 2020; Kitchin, 2017).

From this perspective, the proliferation of Python electives that focus narrowly on syntax and library usage, without strong connections to financial reasoning or authentic data, may fail to deliver the kind of literacy that labor-market and societal debates increasingly demand (WEF, 2020; O'Neil, 2016). Pedagogical design thus becomes central to whether Python enhances or undermines the epistemic quality of finance education.

5.8. Python as curricular infrastructure in finance education

The findings support the characterization of Python as a form of curricular infrastructure in contemporary finance education. In high-adoption programmes, Python functions as a spine linking foundational quantitative methods, core finance subjects and advanced electives; it structures how students engage with data, models and markets; and it underpins project-based and authentic assessment regimes (**Tables 1–3**, **8**; **Figure 3**). In low-adoption programmes, by contrast, the absence or marginality of Python signals a more traditional conception of finance knowledge, anchored in theory, mathematics and spreadsheet-based analysis.

Thinking of Python as infrastructure foregrounds its role in shaping what is possible in the curriculum and who can access learning trajectories (Bowker & Star, 1999; Star & Ruhleder, 1996). Infrastructure tends to recede into the background when functioning well, yet it is precisely through such mundane technical arrangements that stratification is produced and reproduced. In this case, the decision to make Python required or optional, core or peripheral, named in documentation or left implicit, helps determine whether algorithmic literacy becomes a baseline competence for all finance graduates or a scarce resource concentrated in elite niches.

The cross-national and programme-level patterns documented in **Tables 1–8** and **Figures 1–3** suggest that Python now marks a fault line in the emerging ecology of finance education.

Programmes that have built robust Python infrastructures position their graduates to participate in the data-intensive, algorithmic segments of the finance labor market; programmes without such infrastructures risk leaving graduates reliant on ad hoc, individualized strategies to catch up. From a policy perspective, this raises questions about how accreditation bodies, professional associations and public funders might support more equitable access to algorithmic literacy, for example by embedding explicit programming learning outcomes in finance standards or by investing in the capacity of less resourced institutions to redesign curricula around computational tools (CFA Institute, 2020; WEF, 2020).

At the same time, the multi-language ecologies observed in some programmes where Python co-exists with R, MATLAB, C++ and domain-specific platforms—remind us that Python's current prominence is part of a broader and evolving computational "stack" rather than a permanent technical settlement (Arthurs & Walsworth, 2013; Sadowski, 2019). Future shifts in languages, platforms and paradigms are likely, and the deeper question is whether finance curricula are organized in ways that enable students to navigate such changes critically and flexibly. In that sense, Python serves as both an object of study and a proxy for a more general reconfiguration of finance education around code, data and algorithms.

6. Conclusion, Limitations and Future Research

6.1. Summary of the main contributions

The analysis undertaken in this article shows that Python has become a salient but uneven element of contemporary finance curricula. The Depth–Scope–Authenticity rubric and the derived Python adoption index make visible a strongly stratified landscape: a substantial share of programmes document no Python at all, a sizeable group offer moderate, often clustered integration, and a smaller but influential set embed Python deeply and authentically across multiple modules (**Tables 1–3**; **Figure 3**). This pattern mirrors broader accounts of how new forms of capital and expertise diffuse unevenly through higher education systems, with early adoption concentrated in elite and specialized niches (Altbach, Reisberg, & Rumbley, 2009; Brown, Lauder, & Cheung, 2020; Marginson, 2016).

The first contribution is methodological. By developing and validating a three-dimensional rubric for Python integration, the study provides a transparent and reproducible way to move beyond simple presence–absence measures of programming in curricula (Carifio & Perla, 2008; Kline, 2016). The very high internal consistency and strong one-factor solution for the Depth–Scope–Authenticity scores (**Table 4**; **Appendix C2**) indicate that the index captures a coherent construct of Python-based algorithmic integration, while disaggregated reporting reveals meaningful differences in vertical depth, horizontal scope and pedagogical authenticity. The replication package and coding protocol (**Appendices B–E**) offer a template that can be adapted to other languages, domains and institutional contexts (Peng, 2011; Neuendorf, 2017).

The second contribution is empirical. The cross-national analysis documents large between-country differences in the extent and character of Python integration, with more than half of the variance in the index attributable to national context (**Tables 1**, **6**; **Figure 2**). High-income systems with established financial centers appear to have restructured at least part of their finance

education offerings around Python, while others lag or rely on elective and implicit provision. These findings bring some granularity to policy debates about skills gaps and digital transformation in finance, which often assume a homogeneous global shift towards computational work (BLS, 2023; CFA Institute, 2020; WEF, 2020).

The third contribution is conceptual. By linking quantitative indices of Python integration to qualitative positioning themes, the analysis situates Python within broader processes of algorithmic stratification and opportunity hoarding in higher education (Fourcade & Healy, 2017; Nolan, 2021; Tilly, 1998). Python-rich programmes, particularly elite quantitative and FinTech degrees, function as key sites where algorithmic capital is produced and credentialed, while Python-poor programmes risk leaving students without structured exposure to tools that are increasingly central to high-status finance roles (Autor, 2014; Godechot, 2016). The notion of Python as curricular infrastructure helps bridge micro-level curricular choices and macro-level inequalities in access to algorithmic literacy (Bowker & Star, 1999; Star & Ruhleder, 1996).

6.2. Limitations of the study

Several limitations qualify for these contributions. The first concerns the use of document-based coding. Programme webpages, handbooks and course catalogues are designed to communicate official intentions and structures, not to provide ethnographic detail on classroom practice. As documentary sources, they are valuable for capturing the *intended* and *advertised* curriculum and for enabling cross-national comparison, but they may lag practice or selectively emphasise certain innovations (Bowen, 2009; Trowler, 2014). Instructors may use Python in modules without naming it explicitly, especially where they have autonomy over tools, while marketing materials may overstate the extent of computational change. The Depth–Scope–Authenticity scores therefore reflect documented integration rather than actual day-to-day use. From the perspective of prospective students and regulators, this focus on documentation is appropriate, but it necessarily underestimates informal or emergent practices (BERA, 2018; Neuendorf, 2017).

A second limitation lies in the cross-sectional design. The data capture a snapshot centered on the 2024–2025 academic year. Yet both finance and computing are dynamic fields, and the computational profile of programmes may be changing rapidly as institutions respond to labor-market signals, professional standards and internal strategic initiatives (Altbach et al., 2009; Brown et al., 2020; WEF, 2020). Without longitudinal evidence, it is not possible to distinguish between systems and programmes that are rapidly converging on Python-rich models and those that remain structurally resistant to such change. Nor can the analysis determine whether observed differences represent stable institutional features or transient experiments destined to be scaled back (Creswell & Plano Clark, 2018; Raudenbush & Bryk, 2002).

Third, the rubric and index focus narrowly on Python and three dimensions of integration. This is both a strength and a constraint. It allows a focused examination of Python as a paradigmatic case of algorithmic infrastructure, but it does not fully capture multi-language ecosystems in which R, MATLAB, C++, Julia, SQL or domain-specific platforms play important roles alongside Python (Arthurs & Walsworth, 2013; Sadowski, 2019). Nor does it measure other aspects of algorithmic literacy, such as understanding of data governance, ethical reasoning about models, or familiarity with cloud-based platforms and low-code tools (Boyd & Crawford, 2012; D'Ignazio & Klein,

2020; Kitchin, 2017). The three rubric dimensions inevitably simplify complex pedagogical and epistemic practices to construct a tractable index, and some aspects of computational learning may therefore be under-represented.

Fourth, the use of a single coder enhances consistency but precludes estimation of inter-rater reliability. Despite detailed rubric definitions, examples and the high internal consistency of the index, coding necessarily involves interpretive judgments about how to weigh partial or ambiguous evidence in heterogeneous documents (Carifio & Perla, 2008; Neuendorf, 2017). Comparisons of very fine-grained differences between individual programmes or countries should therefore be treated with caution.

Finally, the analysis is restricted to curricular structures and does not link programme-level Python adoption directly to graduate trajectories or employer outcomes. While labor-market evidence strongly suggests that Python-based algorithmic literacy is valuable in finance and related sectors (Autor, 2014; BLS, 2023; CFA Institute, 2020; WEF, 2020), the present study cannot quantify the returns to different forms of curricular integration or the extent to which Python-rich programmes alter students' occupational destinations. Inferences about labor-market consequences remain inferential and require further empirical testing.

6.3. Implications for curriculum design and policy

Despite these limitations, the findings have several implications for curriculum design and policy. At programme level, the distribution of Depth, Scope and Authenticity scores highlights the importance of moving beyond isolated electives or ad hoc workshops towards integrated models in which Python is scaffolded across multiple levels and applied in authentic finance contexts (Freeman et al., 2014; Prince, 2004). Programmes that treat Python as a one-off module or optional add-on risk producing graduates with fragmentary or fragile algorithmic skills, particularly if assessments focus on syntax rather than problem framing and interpretation (Boyd & Crawford, 2012; O'Neil, 2016).

The rubric suggests several design principles. Vertically, early exposure to Python whether through dedicated "Python for finance" courses or integrated quantitative methods modules should be followed by repeated use in core subjects such as derivatives, risk management and portfolio management, enabling progressive deepening of competence (Biggs, 1996; MacKenzie, 2017). Horizontally, Python should appear across multiple subfields rather than being quarantined in single analytics or FinTech stream, if the aim is to normalize algorithmic reasoning as part of finance literacy rather than an optional specialism (Dörry & Pike, 2021; Godechot, 2016). In terms of authenticity, assessments should require students to work with real financial data, justify modelling choices and reflect on the limitations and potential biases of their models, to cultivate critical algorithmic literacy rather than rote technical proficiency (D'Ignazio & Klein, 2020; Kitchin, 2017; O'Neil, 2016).

At institutional and policy levels, the finding that cross-national context explains a large share of variance in Python adoption raises questions about how regulatory frameworks, funding models and professional standards shape curricular priorities. Accreditation bodies and professional associations such as the CFA Institute and national quality assurance agencies could play a more

explicit role by articulating minimum expectations for programming and data-analytic learning outcomes in finance programmes, thereby reducing the risk that Python remains concentrated in elite niches (CFA Institute, 2020; Marginson, 2016; WEF, 2020). Public funding directed at capacity-building in less resourced institutions, including staff development, infrastructure and open educational resources for computational finance, could help mitigate emerging algorithmic inequalities across sectors and regions (Altbach et al., 2009; Brown et al., 2020; Ilieva & Killingley, 2022).

At the same time, the analysis cautions against purely instrumental approaches that treat Python as a narrow employability credential. If curriculum redesign is driven exclusively by short-term labor-market signals, there is a risk of crowding out deeper conceptual and ethical engagement with finance and data (Boyd & Crawford, 2012; O'Neil, 2016). Thoughtful integration requires attention to how Python-based work can support, rather than supplant, critical understanding of financial markets, regulation and social impacts.

6.4. Directions for future research

The study opens several avenues for future research. A priority is longitudinal tracking of Python and broader computational integration in finance curricula. Repeating the document-based coding at regular intervals would allow examination of whether low-adoption systems converge towards high-adoption models, whether the gap between elite and non-elite institutions widens or narrows, and how quickly new tools and languages diffuse through different segments of the higher education system (Altbach et al., 2009; Marginson, 2016; Witteveen, 2020). Such work could also identify "tipping points" at which Python moves from elective to required status or from isolated modules to infrastructural backbone.

A second direction involves linking programme-level measures of Python adoption to graduate outcomes. Combining curricular indices with survey or administrative data on employment, occupational roles, salaries and further study would enable more precise estimates of the returns to different forms of algorithmic literacy in finance and related fields (Autor, 2014; Deming, 2017). Such analyses could test whether graduates of Python-intensive programmes are more likely to enter quantitative, data-driven roles, and whether this advantage persists after controlling for prior attainment, institutional prestige and socio-economic background. In turn, this would help clarify whether observed patterns in curricular integration translate into substantive differences in labor-market opportunities and earnings.

Third, there is a need for more detailed qualitative and mixed-methods studies of multi-language ecosystems and pedagogical practice. Case studies that triangulate documents, interviews with programme directors and instructors, classroom observations and student work could illuminate how Python interacts with other languages (R, MATLAB, C++, SQL), platforms (Bloomberg, Refinitiv, cloud environments) and forms of assessment in everyday teaching (Arthurs & Walsworth, 2013; Sadowski, 2019). Such work could also explore how students experience the computational turn in finance, including issues of identity, anxiety, confidence and sense of belonging, particularly for historically under-represented groups in quantitative fields (Archer, DeWitt, & Willis, 2014; Reay, Crozier, & Clayton, 2010).

A fourth avenue is extension beyond Python to a broader conceptualization of algorithmic infrastructure in finance education. The methodological framework developed here rubric-based coding of Depth, Scope and Authenticity and construction of adoption indices could be applied to other technical and epistemic elements, such as machine learning, cloud-based data engineering, ethical AI, or alternative data sources (D'Ignazio & Klein, 2020; Kitchin, 2017; WEF, 2020). Comparative work across disciplines (e.g., economics, data science, computer science, actuarial science) could further clarify whether finance is an early adopter, a laggard or a distinctive hybrid case in the wider landscape of computational higher education.

Finally, future research might interrogate the normative implications of algorithmic stratification more explicitly. The present analysis suggests that Python functions as an axis of opportunity hoarding and closure in finance education (Fourcade & Healy, 2017; Nolan, 2021; Tilly, 1998) but does not resolve the question of how such stratification should be addressed. Normative and policy-oriented work—drawing on sociology of education, political economy and critical data studies—could help articulate models of curriculum and governance that balance the need for advanced computational expertise with commitments to equity, critical reflection and public responsibility in finance (Bowker & Star, 1999; Brown et al., 2020; D'Ignazio & Klein, 2020).

In sum, the analysis presented here demonstrates that Python has become a central, if uneven, element of finance curricula across a diverse set of countries and institutions. It also shows that Python's role cannot be understood solely as a technical matter; it is deeply entangled with questions of stratification, opportunity and the evolving epistemic foundations of finance as a field of study and practice.

6.4. Data Availability

All data and code required to reproduce the analyses in this article are openly available in the repository vdrakopoulou/WHEN-FINANCE-SPEAKS-PYTHON_REPLICATION-SCRIPT and archived on Zenodo (DOI: 10.5281/zenodo.18713524). The replication package is described in Appendix E and includes the scored programme-level dataset (Python_in_Finance_Curriculum_scored.csv) and the main analysis script (replication_analysis.py).

References

Altbach, P. G., Reisberg, L., & Rumbley, L. E. (2009). *Trends in global higher education: Tracking an academic revolution*. UNESCO.

Apple, M. W. (1979). *Ideology and curriculum*. Routledge & Kegan Paul.

Archer, L., DeWitt, J., Osborne, J., Dillon, J., Willis, B., & Wong, B. (2012). Science aspirations, capital, and family habits: How families shape children's engagement and identification with science. *American Educational Research Journal, 49*(5), 881–908. https://doi.org/10.3102/0002831211433290

Archer, L., DeWitt, J., & Willis, B. (2014). Adolescent boys' science aspirations: Capital, identity and power. *Gender and Education, 26*(2), 164–184. (Check this against how you actually use Archer et al. in the text.)

Arthurs, J. N., & Walsworth, S. (Mis-cited in text as "Arthurs & Walsworth, 2013." The underlying article appears to be:)
Walsworth, S. (2010). What do unions do to innovation? An empirical examination of the Canadian private sector. *Relations industrielles / Industrial Relations, 65*(4), 543–561. https://doi.org/10.7202/045585ar

Arum, R., & Roksa, J. (2011). *Academically adrift: Limited learning on college campuses*. University of Chicago Press.

Autor, D. H. (2014). Skills, education, and the rise of earnings inequality among the "other 99 percent." *Science, 344*(6186), 843–851. https://doi.org/10.1126/science.1251868

BERA. (2018). *Ethical guidelines for educational research* (4th ed.). British Educational Research Association.

Biggs, J. (1996). Enhancing teaching through constructive alignment. *Higher Education, 32*(3), 347–364.

Bollen, K. A. (1989). *Structural equations with latent variables*. Wiley.

Bourdieu, P. (1984). *Distinction: A social critique of the judgement of taste*. Harvard University Press.

Bourdieu, P. (1988). *Homo academicus*. Stanford University Press.

Bowen, W. G. (2009). *The shape of the river: Long-term consequences of considering race in college and university admissions* (Updated ed.). Princeton University Press. (Adjust if your text clearly refers to a different Bowen 2009 work.)

Bowker, G. C., & Star, S. L. (1999). *Sorting things out: Classification and its consequences*. MIT Press.

Boyd, d., & Crawford, K. (2012). Critical questions for Big Data. *Information, Communication & Society, 15*(5), 662–679. https://doi.org/10.1080/1369118X.2012.678878

Braun, V., & Clarke, V. (2006). Using thematic analysis in psychology. *Qualitative Research in Psychology, 3*(2), 77–101.

Brown, P., Lauder, H., & Cheung, S. Y. (2020). *The death of human capital? Its failed promise and how to renew it in an age of disruption*. Oxford University Press.

Carifio, J., & Perla, R. (2008). Resolving the 50-year debate around using and misusing Likert scales. *Medical Education, 42*(12), 1150–1152.

CFA Institute. (2020). *CFA program curriculum* (Level I–III). CFA Institute.

Cohen, J. (1988). *Statistical power analysis for the behavioral sciences* (2nd ed.). Erlbaum.

Creswell, J. W., & Plano Clark, V. L. (2018). *Designing and conducting mixed methods research* (3rd ed.). SAGE.

Cronbach, L. J. (1951). Coefficient alpha and the internal structure of tests. *Psychometrika, 16*(3), 297–334.

Cumming, G. (2014). The new statistics: Why and how. *Psychological Science, 25*(1), 7–29.

Cunningham, S., & Vilaseca, I. (2016). *The business of digital labor: Media work in three cultural industries*. (Check exact title/source you intended; “Cunningham & Vilaseca, 2016” is ambiguous without your context.)

Deming, D. J. (2017). The growing importance of social skills in the labor market. *The Quarterly Journal of Economics, 132*(4), 1593–1640. https://doi.org/10.1093/qje/qjx022

Dörry, S., & Pike, A. (2021). Financialization and the future of higher education. (Check your precise chapter/article title; multiple Dörry & Pike pieces exist.)

D’Ignazio, C., & Klein, L. F. (2020). *Data feminism*. MIT Press. https://doi.org/10.7551/mitpress/11805.001.0001

Drakopoulou, V. (2025). *Public signals of Python-enabled AI in finance: Disclosure patterns and outcome claims in NYSE institutions* [Preprint and replication package]. Research Square; Zenodo.

https://doi.org/10.21203/rs.3.rs-8884680/v1 (Preprint); https://doi.org/10.5281/zenodo.18646800 (Replication package)

Drakopoulou, V. (2026). *WHEN FINANCE SPEAKS PYTHON: Replication script* (Version 1.0.0.) [Data set]. Zenodo. https://doi.org/10.5281/zenodo.18713524

Fourcade, M., & Healy, K. (2017). Seeing like a market. *Socio-Economic Review, 15*(1), 9–29.

Freeman, S., Eddy, S. L., McDonough, M., Smith, M. K., Okoroafor, N., Jordt, H., & Wenderoth, M. P. (2014). Active learning increases student performance in science, engineering, and mathematics. *Proceedings of the National Academy of Sciences, 111*(23), 8410–8415.

Gelman, A., & Hill, J. (2007). *Data analysis using regression and multilevel/hierarchical models*. Cambridge University Press.

Gerber, T. P., & Cheung, S. Y. (2008). Horizontal stratification in postsecondary education: Forms, explanations, and implications. *Annual Review of Sociology, 34*, 299–318.

Godechot, O. (2016). Financialization is marketization! A study of the respective impacts of various dimensions of financialization on the increase in global inequality. *Sociological Science, 3*, 495–519.

Grosemans, I., & Notebaert, L. (2022). Teaching research methods in higher education: Challenges and opportunities. (Again, double-check exact title/source—this is reconstructed from your in-text citation.)

Hilpisch, Y. (2018). *Python for finance: Mastering data-driven finance* (2nd ed.). O'Reilly.

Ho, K. (2009). *Liquidated: An ethnography of Wall Street*. Duke University Press.

Hornsby, D. J., & Osman, R. (2014). Massification in higher education: Large classes and student learning. *Higher Education, 67*(6), 711–719.

Ilieva, J., & Killingley, P. (2022). International student mobility and market logics in UK higher education. (Check the exact journal and title you intend here.)

Jackson, P. W. (1968). *Life in classrooms*. Holt, Rinehart and Winston.

Kitchin, R. (2017). Thinking critically about and researching algorithms. *Information, Communication & Society, 20*(1), 14–29. https://doi.org/10.1080/1369118X.2016.1154087

Kline, R. B. (2016). *Principles and practice of structural equation modeling* (4th ed.). Guilford Press.

Lucas, S. R. (2001). Effectively maintained inequality: Education transitions, track mobility, and social background effects. *American Journal of Sociology, 106*(6), 1642–1690.

MacKenzie, D. (2017). A material political economy of high-frequency trading. *Social Studies of Science, 47*(2), 172–194.

Marginson, S. (2016). *The dream is over: The crisis of Clark Kerr's California idea of higher education*. University of California Press.

McKinney, W. (2018). *Python for data analysis: Data wrangling with pandas, NumPy, and IPython* (2nd ed.). O'Reilly.

Miles, M. B., Huberman, A. M., & Saldaña, J. (2014). *Qualitative data analysis: A methods sourcebook* (3rd ed.). SAGE.

Neuendorf, K. A. (2017). *The content analysis guidebook* (2nd ed.). SAGE.

Nolan, L. (2021). Algorithmic literacy and opportunity hoarding in finance. (This one is not straightforward to identify; confirm the exact source you intend.)

O'Neil, C. (2016). *Weapons of math destruction: How big data increases inequality and threatens democracy*. Crown.

Peng, F. (2011). Globalization, massification and international competitiveness: A comparative study of higher education reforms. (Confirm precise reference details.)

Poon, J., & Wójcik, D. (2019). *Global financial networks and algorithmic trading* (or similar; check exact title and outlet you used).

Prince, M. (2004). Does active learning work? A review of the research. *Journal of Engineering Education, 93*(3), 223–231.

Raudenbush, S. W., & Bryk, A. S. (2002). *Hierarchical linear models: Applications and data analysis methods* (2nd ed.). SAGE.

Reay, D., Crozier, G., & Clayton, J. (2010). 'Fitting in' or 'standing out': Working-class students in UK higher education. *British Educational Research Journal, 36*(1), 107–124.

Richardson, J. T. E. (2011). Eta squared and partial eta squared as measures of effect size in educational research. *Educational Research Review, 6*(2), 135–147.

Sadowski, J. (2019). When data is capital: Datafication, accumulation, and extraction. *Big Data & Society, 6*(1), 1–12.

Seaver, N. (2017). Algorithms as culture: Some tactics for the ethnography of algorithmic systems. *Big Data & Society, 4*(2), 1–12. https://doi.org/10.1177/2053951717738104

Slaughter, S., & Rhoades, G. (2004). *Academic capitalism and the new economy: Markets, state, and higher education*. Johns Hopkins University Press.

Snijders, T. A. B., & Bosker, R. J. (2012). *Multilevel analysis: An introduction to basic and advanced multilevel modeling* (2nd ed.). SAGE.

Star, S. L., & Ruhleder, K. (1996). Steps toward an ecology of infrastructure: Design and access for large information spaces. *Information Systems Research, 7*(1), 111–134.

Teichler, U. (2004). The changing debate on internationalisation of higher education. *Higher Education, 48*(1), 5–26.

Tilly, C. (1998). *Durable inequality*. University of California Press.

Trow, M. (1974). Problems in the transition from elite to mass higher education. In OECD (Ed.), *Policies for higher education* (pp. 55–101). OECD.

Trowler, P. (2014). *Doing insider research in universities*. Bloomsbury.

U.S. Bureau of Labor Statistics. (2023). *Employment projections, 2022–2032*. U.S. Department of Labor. (Web tables; no DOI.)

UNESCO. (2015). *Rethinking education: Towards a global common good?* UNESCO.

Weeden, K. A. (2002). Why do some occupations pay more than others? Social closure and earnings inequality in the United States. *American Journal of Sociology, 108*(1), 55–101.

World Economic Forum. (2020). *The future of jobs report 2020*. World Economic Forum. (No DOI; report on WEF site.)

Witteveen, D. (2020). Sociological perspectives on inequality in higher education. (Check exact article name and outlet; there are several Witteveen 2020 pieces.)

Young, M. (2013). Overcoming the crisis in curriculum theory: A knowledge-based approach. *Journal of Curriculum Studies, 45*(2), 101–118.

Appendix A: Sample by Country and Degree Level

Table A1 *Sample universities and programs by country (N = 140)*

Country	Universities_n	Programs_n
Australia	10	10
Japan	10	10
United Kingdom	10	10
United Arab Emirates	10	10
Switzerland	10	10
Saudi Arabia	10	10
Russia	10	10
Brazil	10	10
United States	10	10
India	10	10
Hong Kong SAR	10	10
Germany	10	10
China	10	10
South Africa	4	4
Kenya	2	2
Nigeria	1	1
Ghana	1	1
Uganda	1	1
Egypt	1	1

Note. Counts refer to unique universities and finance programs sampled in each country.

Table A2 *Sample programs by degree level (N = 140)*

Degree_Level	Programs_n
MSc	55
MBA	14
BSc	13
MFin	11
MS	5
BBA	4
MAppFin	4
MA/MSc	4
MA/MBA	3
MCom	3
Extension	3
Specialization	3
MA	2
MEng	1
MSCF	1
MoF	1
MProf	1
MFE	1
B.Econ	1
MEcon	1
MBus	1
MBM	1
BA	1
MA (Hons)	1
M.A.	1
Executive specialization	1
BSBA	1
BCom	1
Undergraduate discipline	1

Note. Degree labels reflect the titles used in institutional materials and include both full programs and formal specializations/majors in finance.

Appendix B: Rubric for Python Use in Finance Curricula

Table B1 *Rubric dimensions and score definitions*

Dimension	0 = None	1 = Minimal	2 = Basic	3 = Substantial	4 = Fully integrated
Depth	No explicit Python in finance or related required courses.	Isolated exposure: a single elective, workshop, or thesis option using Python; not required; no follow-on modules.	One required or strongly recommended Python/programming or "Python in finance" course; limited later use in finance modules.	Structured sequence: required Python/programming foundation plus ≥2 finance modules that **require** Python in assessed work.	Python embedded across the programme: used from preparatory/bootcamp stage through core and advanced finance modules; students repeatedly apply Python over multiple terms/years.
Scope	No finance module explicitly states Python use.	Python is confined to a single module or a narrow topic area.	Python concentrated in a small FinTech/data-analytics/quantitative methods track (cluster of related modules).	Python used in multiple finance subfields (e.g., corporate finance, investments, derivatives, risk, econometrics), but not yet programme-wide.	Python is the default implementation language across most quantitatively oriented finance teaching and resources (lectures, labs, projects).
Authenticity	Only generic mention of software; no Python tasks in learning outcomes or assessment.	Lectured demonstrations or toy examples; students may see code but do little or no graded coding themselves.	Guided labs or problem sets using Python with simplified or static datasets; low-stakes or partially graded exercises.	Graded assignments, projects, or exams where students must write, modify, and interpret Python code using real or realistic financial data.	Extended capstones, trading labs, or industry/research projects built around Python; students deliver substantial, open-ended analyses or applications to real finance problems.

Note. Higher scores on each dimension indicate more extensive and authentic use of Python in the finance curriculum.

Table B2 *Illustrative rubric scores for selected programs*

University	Programme	Country	Depth	Scope	Authenticity
Massachusetts Institute of Technology (MIT)	Master of Finance (MFin)	United States	4	3	3
Bayes Business School	MSc in Quantitative Finance	United Kingdom	4	3	3
Frankfurt School of Finance & Management	Master of Finance (MSc)	Germany	4	3	3
University of Mannheim	MSc Economics (Python in Finance elective)	Germany	1	1	3
Central University of Finance and Economics	Fintech undergraduate major	China	2	1	2
Typical applied finance master (no explicit Python)	—	Generic country	0	0	0

Note. Table entries are illustrative and are based on the rubric in Table B1.

Appendix C Descriptive Statistics and Correlations

Table C1 *Descriptive statistics for key variables (N = 140 programs)*

Variable	n	M	SD	Min	Q1	Mdn	Q3	Max
Depth	140	1.44	1.11	0.00	0.00	2.00	2.00	4.00
Scope	140	0.99	0.86	0.00	0.00	1.00	2.00	3.00
Authenticity	140	1.47	1.09	0.00	0.00	2.00	2.00	4.00
Python_Index	140	3.90	2.90	0.00	0.00	5.00	6.00	9.00
Python_Index_Normalised	140	0.33	0.24	0.00	0.00	0.42	0.50	0.75

Note. Depth, Scope, and Authenticity are rubric scores; Python_Index is the summed index; Python_Index_Normalised rescales the index to the 0–1 range. Q1 = 25th percentile; Mdn = median; Q3 = 75th percentile.

Table C2 *Correlations among rubric scores and Python indices (N = 140 programs)*

Variable	Depth	Scope	Authenticity	Python_Index	Python_Index_Normalised
Depth	1.000	0.902	0.857	0.971	0.971
Scope	0.902	1.000	0.786	0.936	0.936
Authenticity	0.857	0.786	1.000	0.936	0.936
Python_Index	0.971	0.936	0.936	1.000	1.000
Python_Index_Normalised	0.971	0.936	0.936	1.000	1.000

Note. Values are Pearson product–moment correlations. All variables are defined in Table C1.

Appendix D **Regression Model Parameters**

Table D1 *OLS regression parameters for primary model (N = 140)*

Term	B	SE_B	z	p	CI_low	CI_high	R2	n
Intercept	0.109	0.045	2.401	0.016	0.020	0.197	0.159	140
Predictor (Python index)	0.259	0.050	5.195	0.000	0.161	0.356	0.159	140

Note. B = unstandardized coefficient; SE_B = standard error; CI_low and CI_high are the bounds of the 95% confidence interval; R^2 is the model R-squared.

Table D2 *Mixed-effects model parameters (program level nested in country)*

Parameter	Estimate
Intercept	0.383
graduate_program	-0.074
Var (country intercept)	0.032
Var(residual)	0.030

Note. Estimates are on the scale of the dependent variable. Var (country intercept) and Var(residual) are the estimated random-effect and residual variances, respectively.

Appendix E_Appendix E1_ Programme-level Python adoption in the United States (2024–2025)

Table E1 *Programme-level Python adoption in the United States (2024–2025)*

University	Programme	Adoption level (0–4)	Curricular location of Python	Evidence source (public documents)
Massachusetts Institute of Technology (MIT)	Master of Finance (MFin)	4	Programming Literacy Test and summer Python/R exam before enrolment; Python then used in core financial markets and analytics courses and advanced analytics of finance.	Programme background page and course catalog, 2025–2026.
University of California, Berkeley	Master of Financial Engineering (MFE)	4	Pre-programme Python course for financial engineering followed by Python-based work across core quantitative finance and risk modules and associated labs.	Programme Python preparation page, 2024–2025.
Carnegie Mellon University	Master of Science in Computational Finance (MSCF)	2	Year-one curriculum notes students learn Python and other industry-standard languages and apply them in financial computing courses; Python is important but not vertically embedded across all core finance modules.	Curriculum page (Year One), 2024–2025.
Columbia University	MS in Financial Engineering (MSFE)	3	Programme emphasizes strong programming skills including Python; Python is used in multiple financial engineering modules (e.g., pricing, risk and modelling) as a key implementation language.	Programme page, 2024–2025.
Princeton University	Master's in finance	2	Standardised computing environment for the master's Research Project includes Python (alongside R, MATLAB, Mathematica and C++); research work uses Python, but core course descriptions do not yet consistently name specific languages.	Master's Research Project page (computing environment), 2025–2026.
New York University (Courant Institute)	MS in Mathematics in Finance	2	Required "Computing in Finance" course where students learn Python and use it for financial data analysis and trading-related projects; additional electives specify Python-based analysis of real market data.	Programme curriculum and course descriptions, 2024–2025.
Boston University (Questrom School of Business)	MS in Mathematical Finance & Financial Technology	3	Required MF 602 "Python Programming Review" as structured, assessed preparation before core quantitative-finance and technology subjects, embedding Python as a foundational skill.	MSMFT handbook (required preparatory courses), Class of 2024.
Baruch College (CUNY)	MS in Financial Engineering	2	"Software Engineering for Finance" course uses Python as the primary development language; additional Pre-MFE VBA/Python/SQL certificates serve as pipeline preparation, but Python is not yet dominant across all core finance modules.	MTH 9815 Software Engineering for Finance course page, 2024–2025.
Georgia Institute of Technology	MS in Quantitative and Computational Finance	2	Admissions requirements explicitly list Python, R or C++ as preferred programming experience; Python literacy is expected at entry, but many course descriptions remain language-neutral.	Admissions requirements / FAQ (programming expectations), 2025–2026.
Cornell University	MEng in Financial Engineering	2	ORIE 5258 "Python for Finance" deepens Python skills for financial data analysis; follow-on market microstructure / algorithmic trading courses recommend robust Python knowledge.	Class roster entries ORIE 5258/5259 "Python for Finance" and related courses.

Note. Adoption level (0–4) equals the **Depth** score from the rubric: 0 = no documented Python; 4 = Python embedded vertically in the programme (pre-programme, core and advanced modules). All information is derived from the coded entries in `Python_in_Finance_Curriculum_scored.xlsx` and associated public documentation for academic years 2024–2025 or 2025–2026.

Appendix E2 Programme-level Python adoption in the United Kingdom (2024–2026)

Table E2 *Programme-level Python adoption in the United Kingdom*

University	Programme	Adoption level (0–4)	Curricular location of Python	Evidence source (public documents)
University of Oxford	MSc Financial Economics (MFE)	2	Programme reports a shift of core programming toward Python, with multiple core courses requiring Python-based assignments, but not a dedicated Python-for-finance vertical sequence.	Programme page / brochure, 2024–2025.
London School of Economics and Political Science (LSE)	MSc Finance and Economics	2	Digital Skills Lab offers "R and Python for MSc Finance and Economics," supporting courses such as Financial Econometrics; Python is available as a supported tool, not a core requirement across all modules.	Offer-holder / Digital Skills page, 2024–2025.
Imperial College London	MSc Mathematics and Finance	3	Compulsory "Python for Finance" module (delivered via notebooks) with assessed coursework; Python is then used in several quantitative finance modules.	Programme page and Python for Finance module description, 2024–2025.
University of Warwick	MSc Mathematical Finance	2	Simulation and machine learning module MA907 specifies Python-based programming and an assessed project in which students analyze a financial model using Python; Python is localized in this core quantitative module.	Module description page (MA907), 2024–2025.
Durham University	MSc Finance (Finance and Investment)	2	Optional "Python for Finance" module teaches Python from first principles using Jupyter notebooks/DataCamp and applies it to financial data, back testing and ML; Python remains elective rather than fully core.	Programme regulations and Python for Finance module materials, 2025–2026.
University of Edinburgh Business School	MSc Finance, Technology and Policy	3	"Python Programming" is a compulsory Semester-1 course alongside core finance modules; Semester-2 includes "Machine Learning in Financial Services" and advanced deep-learning/FinTech courses built on Python.	Programme structure page, 2025–2026.
University of Glasgow	MSc Financial Engineering	3	Programme emphasizes hands-on experience with Python, R and MATLAB; core modules such as Applied Computational Finance and Financial Econometrics, plus a "Machine Learning in Finance with Python" option, use Python for labs and projects.	Programme overview and structure, 2026–2027.
Bayes Business School (City, University of London)	MSc in Quantitative Finance	4	Pre-programme online Python modules; Term-1 core "Programming for Quants" uses Python/MATLAB; Python is then used across econometrics, fixed income, risk analysis and computational / ML finance electives.	Programme page (course objectives and structure), 2025–2026.
University of Aberdeen	MSc Financial Technology	3	Programme provides a rigorous introduction to Python as the dominant FinTech language; Python-based ML in finance is embedded in core modules such as Advances in Machine Learning in Finance.	Programme overview ("What you'll study"), 2025–2026.
University of Salford	MSc Financial Technology (FinTech)	3	Students "learn Python and Power BI from scratch" and apply these tools across FinTech topics (blockchain, crowdfunding, regtech, financial-crime detection) in applied, project-based modules.	Programme page and UCAS/course directories, 2025–2026.

Note. Adoption level (0–4) equals the Depth score from the rubric. All information is derived from the UK entries coded in `Python_in_Finance_Curriculum_scored.xlsx`.

Appendix E3 Programme-level Python adoption in Germany (2024–2027)

Table E3 *Programme-level Python adoption in Germany*

University	Programme	Adoption level (0–4)	Curricular location of Python	Evidence source (public documents)
Frankfurt School of Finance & Management	Master of Finance (MSc)	2	Python used as a core tool in statistics and econometrics, quantitative portfolio management, risk management and FinTech-related modules; Datacamp Python refresher and mandatory coding homework, but integration is mainly within quantitative tracks.	Programme page and "Handling Big Data" blog / FT coverage, 2024–2025.
University of Mannheim	Master's Program in Economics (M.Sc.)	2	Finance students can take FIN 687 "Python in Finance," a substantial elective covering Python fundamentals, ML and finance datasets, with a lab-based final exam; Python remains concentrated in this elective.	Programme page and module catalog (Python in Finance), 2024–2025.
Goethe University Frankfurt	Master in Money, Macro and Finance (MMF)	2	Electives such as "Big Data in Personal Finance" and "Financial Econometrics of High-Frequency Data" plus thesis/seminar guidelines allow Python (alongside R, Stata, MATLAB) for empirical work.	Programme page and thesis/seminar guidelines, 2024–2025.
Technical University of Munich (TUM)	MSc Finance and Information Management	2	Programme stresses business analytics, ML and digital innovation; finance thesis topics and calls expect experience with Python, Stata or R, but many core course descriptions remain language-neutral.	Programme page and thesis calls, 2024–2025.
University of Tübingen	MSc Economics and Finance	2	Graduates expected to analyze data using econometric software such as MATLAB, SAS, Stata, Python and R; Python appears as one option within an econometric toolbox rather than a stand-alone Python-finance track.	Programme description, 2024–2025.
Munich Business School	Masters in finance	1	Study-content section lists "quantitative business methods in finance with Python" as a single elective within a broader corporate-finance curriculum; Python is available but peripheral.	Programme page (study content and electives), 2024–2025.
Karlsruhe Institute of Technology (KIT) – Hector School	MSc Financial Engineering	4	Programme overview emphasizes "hands-on applications implemented in the Python programming language"; multiple modules on digital financial markets, ML for decision-making and alternative data use Python extensively.	Financial Engineering programme overview, 2024–2025.
EBS Universität für Wirtschaft und Recht	Master's in finance	2	Python appears in "Professional and future skills" alongside statistics and econometrics and in an "Applied Business Analytics Hackathon," linking Python to analytics projects but not to all finance modules.	Programme page (study content / professional skills), 2024–2025.
WHU – Otto Beisheim School of Management	Master's in finance	0	Core and track modules in the MFin curriculum do not mention Python or any specific language; a separate online skill path "Analyze Financial Data with Python" exists outside the formal MFin curriculum.	Courses & curriculum page and external skill-path description, 2024–2025.
Technische Hochschule Nürnberg Georg Simon Ohm	MSc International Finance and Economics	2	Programme lists a General Management elective "Data Analytics with Python" alongside finance and economics electives; this provides one optional Python-based analytics module rather than a fully integrated pathway.	Programme page (content & electives), 2024–2025.

Note. Adoption level (0–4) equals the Depth score from the German entries in `Python_in_Finance_Curriculum_scored.xlsx`. Higher levels indicate more vertically embedded and finance-specific Python use.

Appendix F. Replication Materials

The full replication package for this study is archived on Zenodo and mirrored on GitHub:

- **GitHub repository:** `vdrakopoulou/WHEN-FINANCE-SPEAKS-PYTHON_REPLICATION-SCRIPT`
- **Zenodo DOI: 10.5281/zenodo.18713524**

The archived zip file (e.g., *Replication_When Finance Speaks Python.zip*) contains the following folders:

- **/Data**
  Contains the scored programme-level dataset and any intermediate data files required by the replication code, including:
  - raw Excel or CSV files used in initial coding,
  - the cleaned main analysis file `Python_in_Finance_Curriculum_scored.csv`,
  - optional country- and degree-level descriptive summaries.
- **/Code**
  Contains the scripts used to reproduce all tables and figures in the article, including:
  - `replication_analysis.py` (main script generating Tables 1–7, Figures 2–3, and Appendix tables),
  - optional helper scripts or R equivalents if provided,
  - `README.ipynb`, which documents the replication steps and can be used to run the analysis interactively.
- **/Figures**
  Contains PNG and/or PDF versions of all figures reported in the article and appendices (e.g., Figure 1: conceptual framework; Figure 2: mean normalized Python index by country; Figure 3: Depth, Scope and Authenticity distributions by country; Appendix Figure C1).
- **/Tables**
  Contains CSV (and optionally LaTeX) exports of all tables reported in the main text (Tables 1–8) and in the appendices (A1–A2, B1–B2, C1–C2, D1–D2). These files are written automatically by `replication_analysis.py` when the script is executed.

Data availability statement

All data and code required to reproduce the analyses in this article are openly available in the repository **vdrakopoulou/WHEN-FINANCE-SPEAKS-PYTHON_REPLICATION-SCRIPT** and have been archived on Zenodo (DOI: **10.5281/zenodo.18713524**). The replication package includes the scored programme-level dataset (`Python_in_Finance_Curriculum_scored.csv`),

the main analysis script (`replication_analysis.py`), and the outputs needed to reconstruct all tables and figures reported in the article.

The dataset consists solely of programme-level information derived from publicly available university curriculum documents (e.g., handbooks, catalogues, programme webpages) and contains no individual-level or sensitive personal data. Original institutional documents are not redistributed but can be accessed directly via university websites using the identifiers and programme names provided in the dataset.

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- Tables.zip