

مخاطر بناء تطبيقات بايثون بنفسك: الإنذارات الكاذبة والناشر المجهول وتراخيص الشفيرة

كوري شميدت



المخلص:

يروي هذا المقال تجربة مؤلفه، وهو أرشيفي ذو معرفة بايثون، في بناء تطبيق مكتبي لتبسيط سير عمل تصدير ملفات (EAD.xml) من (ArchivesSpace) وتنظيفها ورفعها إلى موقع أدوات البحث الأرشيفية في مكتبات جامعة جورجيا. يصف بناء واجهة رسومية بمكتبة (PySimpleGUI)، وتحويلها إلى ملف تنفيذي بأداة (PyInstaller)، وإنشاء برنامج تثبيت لويندوز بأداة (Inno Setup). ثم يعرض ثلاث عقبات غير متوقعة: تحذيرات ويندوز من الناشر المجهول وتكلفة توقيع الشيفرة، والإنذارات الكاذبة لبرامج مكافحة الفيروسات وطرق التعامل معها عبر (cx-Freeze) و (VirusTotal) والإبلاغ عنها، وتغيير ترخيص (PySimpleGUI) الذي فرض رسوماً على المطورين في المؤسسات التعليمية. ويخلص إلى أن بناء الثقة مع المستخدمين يتطلب جهداً مقصوداً، وأن على من يسلك هذا الطريق أن ينتبه إلى خطواته.

الكلمات المفتاحية: تطبيقات بايثون، الإنذارات الكاذبة لبرامج مكافحة الفيروسات، توقيع الشيفرة، تراخيص البرمجيات، سير العمل الأرشيفي

يصعب صنع تطبيقات بايثون، ولكن ليس دائماً على النحو الذي تتوقعه. ففي سعيي إلى تبسيط مسارات العمل الأرشيفية لدينا، انطلقت لاكتشاف كيفية صنع تطبيقات مكتبية مستقلة لأرشيفينا ومعالجي المجموعات لدينا، بحيث تصبح مسارات العمل المتكررة أسهل وأوضح للمستخدم. ولأنني آتٍ من خلفية أرشيفية مع بعض المعرفة بايثون (Python)، تعلمت كيف أبرمج أشياء مثل واجهات المستخدم الرسومية (Graphical User Interfaces, GUIs)، وكيف أنشئ ملفات تنفيذية (ثنائية)، وكيف أولّد برامج تثبيت للبرمجيات على ويندوز. وقد برزت عقبات غير متوقعة لا أزال أصادرها: التعامل مع برامج مكافحة الفيروسات التي تصفّ ملفاتك على أنها برمجيات خبيثة، وإطلاق مايكروسوفت ويندوز رسائل تحذير من تنزيل برمجيات من ناشرين مجهولين (وهي محقة في ذلك)، وفكّ

تشابك تغييرات الترخيص في مكتبة بايثون كانت متاحة مجانًا من قبل. وسأشارك في هذا المقال رحلتي في إنشاء تطبيقات قائمة على بايثون لمستخدمينا وتوزيعها والتعامل مع تبعات صنعها، وأقدم نصائح بشأن ما ينبغي الانتباه إليه إن كنت تتطلع إلى فعل شيء مماثل.

مقدمة:

اجعل مسارات العمل أسهل. كان ذلك، ولا يزال، الدافع وراء رحلتي في استكشاف تطوير تطبيقات بايثون. فقد تدربت أرشيفيًا، ولم أكن مطور برمجيات. وعالجت عددًا من المجموعات ورقمتها خلال فترات تدريب عملي في أرشيفات ومكتبات مختلفة، ثم تعرفت إلى بايثون في الدراسات العليا، حيث تعلمنا أساسيات البرمجة وكيفية أتمتة مسارات العمل المتكررة. ومع أنني لم أتوقع استخدام البرمجة خارج الدراسات العليا، فقد اكتشفت فائدتها في وظيفتي الأولى بمكتبات المجموعات الخاصة بجامعة جورجيا (University of Georgia Special Collections Libraries)، حيث أدرت مشروع الانتقال من نظام (Archivists' Toolkit)، الذي كنا نستخدمه لإدارة الأوصاف الأرشيفية لمجموعتنا، إلى (ArchivesSpace).

وكان من العناصر الحاسمة في هذا الانتقال أن تتبين كيف سنعيد صياغة مسار عملنا القائم لتصدير ملفات (EAD.xml)، وإجراء سلسلة من عمليات التنظيف والتوحيد المعياري عليها، ورفعها إلى موقعنا الخاص بأدوات البحث الأرشيفية (finding aids). وقادني ذلك إلى استخدام بايثون لاستبدال مسار عملنا القائم بشيء أسهل في التوزيع وأبسط في الاستخدام لأعضاء هيئة التدريس والموظفين المعنيين بمعالجة المجموعات. فتعلمت كيف أصنع واجهة مستخدم رسومية باستخدام بايثون، وكيف أنشئ ملفًا تنفيذيًا (.exe) يحزم بايثون واعتمادياته في ملف واحد، وكيف أستخدم أداة مجانية لصنع برنامج تثبيت لمايكروسوفت ويندوز. وعلى غير توقع، وجدت نفسي أتحايل على وسائل الحماية التي وضعتها مايكروسوفت نفسها ضد الناشرين المجهولين، وعلى برامج مكافحة الفيروسات التي كانت تصف ملفات التنفيذ على أنها خبيثة، فضلًا عن فك تشابك تغييرات الترخيص الأخيرة في حزمة (PySimpleGUI) التي كنت أستخدمها للتطبيق، والتي كانت من قبل مفتوحة المصدر ومجانية الاستخدام. وقد أجبرتني هذه التحديات على مواجهة معنى بناء الثقة مع المستخدمين، ومعنى صيانة تطبيقات مكتبية مفتوحة المصدر مطورة داخليًا، وكيف أن جعل مسارات العمل أسهل لا يخص المستخدمين وحدهم، بل المطورين أيضًا.

الخلفية:

إن القول بأن مسار عملنا القائم لتصدير ملفات (EAD.xml) وتحريرها ورفعها إلى موقع أدوات البحث كان ثقيلًا هو تهوين من الأمر. فقد كان إخراج البيانات الوصفية لمجموعتنا من نظام إدارة الأرشيفات ووضعها على موقعنا يتطلب تصدير كل مجموعة على حدة يدويًا من (Archivists' Toolkit) في ملفات (EAD2002 XML)، وتشغيل نص برمجي بلغة (Perl) كُتب قبل عقد من الزمن. ولأن النص البرمجي افتقر إلى التوثيق وتسجيل الأخطاء، كان على المستخدمين أن يأملوا ألا تُرجع عملية التنظيف أي أخطاء، لأنهم لم تكن لديهم أدنى فكرة عن كيفية عمله أو كيفية إصلاحه. وإن نجحت عملية التنظيف، كان على المستخدمين أن يفتحوا الطرفية (terminal) أو يستخدموا (WinSCP) لرفع الملف إلى خادم موقع أدوات البحث. ثم كان الأرشيفيون والمعالجون يدخلون يدويًا وسيطة سطر أوامر في دليل محدد على خادم الويب لفهرسة الملفات التي رُفعت للتو إلى الخادم فقط، لكي تصبح مرئية وقابلة للبحث على الموقع. وإن لم تُدخّل هذه الوسيطة تمامًا كما هي مكتوبة في توثيقنا الخاص، فقد تتسبب في أن يفهرس الموقع كل ملف في الدليل، وهي ملفات يزيد مجموعها على 5000 ملف (EAD.xml)، فيصبح الموقع غير قابل للاستخدام فعليًا لباحثينا وروادنا مدة 24 إلى 48 ساعة تالية. وقد تنشأ مشكلات أخرى في العملية أيضًا، مثل

عدم ضبط الأذونات الصحيحة لملفات (lazy)، وهي ملفات تُستخدم للمساعدة في تحميل العناصر في صفحة المجموعات حين يطلبها مستخدم (Under the Hood « XTF). وحين يحدث ذلك، تعرض صفحة المجموعة شاشة خطأ تبدو مستعصية على الفهم، رغم ظهور المجموعة في نتائج البحث لدينا بكل البيانات الوصفية المناسبة. وحين نظرت إلى كل الطرق الممكنة التي قد تسوء بها الأمور أو تصعب بها إدارتها على مستخدمينا، قلت في نفسي: «لا بد أننا نستطيع تبسيط هذه العملية وجعلها أكثر اتساقًا». وهنا خطرت لي بایشون.

بناء تطبيق مكتبي بایشون:

تعلمت أن بایشون رائعة لأشياء كثيرة، لكنها جيدة بوجه خاص في أتمتة العمليات القابلة للتكرار والمتسقة. ولأننا كنا ننتقل إلى (ArchivesSpace)، كنت أعلم أننا نستطيع استخدام واجهة برمجة التطبيقات (API) الخاصة بـ (ArchivesSpace) وبایشون للبحث عن المجموعات وتصدير ملفات (EAD.xml) الخاصة بها فرادى أو على دفعات. ولتنظيف ملفات (EAD.xml)، لدى بایشون مكتبة لذلك: (lxml). فهي تستطيع البحث في ملفات (XML) بحثًا عن وسوم وسمات محددة، وكذلك عن المحتوى، وتحريرها حسب الحاجة. ولرفع الملفات وتحريرها على خادم ملفات، هناك مكتبتا (SCP) و (SFTP) لبایشون، وقد أراني هذا الدليل المفيد كيفية استخدامهما لهذا الغرض بالضبط (SSH & SCP in Python with Paramiko). فشرعت في العمل وبنيت نصوًا برمجية مختلفة لبایشون للتعامل مع تصدير الملفات وتحريرها ورفعها، لكنني كنت بحاجة إلى طريقة تجمع هذه العملية للمستخدم، ويُفضّل أن يكون ذلك دون الحاجة إلى العمل داخل الطرفية.

ثم اكتشفت (PySimpleGUI). وهي مكتبة بایشون مصممة لصنع واجهات المستخدم الرسومية باستخدام بایشون، تستعير مفاهيم من مكتبات بایشون أخرى تؤدي أشياء مماثلة وتضيف إليها ميزات الخاصة. وكانت (PySimpleGUI) تحتوي على كل شيء، من الأزرار إلى مربعات إدخال المستخدم، والنوافذ المنبثقة، والإعدادات الافتراضية، والقوائم المنسدلة، ومربعات الاختيار، وأزرار الاختيار، أي كل ما يمكن أن تربطه تقريبًا بميزات التطبيقات المكتبية. فبدأت تصميم تطبيقي باستخدام مخططات هيكلية بسيطة (wireframes)، ثم اخترت ميزات مختلفة تقدمها (PySimpleGUI) بينما كنت أتبيّن أيها سيجعل كل مهمة أوضح للمستخدم. وبعد شيء من العمل، عرضت الواجهة الرسومية على أعضاء هيئة التدريس والموظفين لدينا، فكانوا متحمسين جدًا. فكل ما كان على المستخدم فعله هو إدخال معرف المجموعة (أو معرفات المجموعات) التي يريد إدخالها في مربع إدخال على الشاشة والنقر على زر «(Export)» [تصدير]. وكان بإمكانه اختيار عمليات التنظيف المحددة التي يريد تشغيلها والمكان الذي توضع فيه الملفات المكتملة. ولم يعد من الضروري إدخال أوامر الطرفية لرفع ملف (EAD.xml)؛ بل أصبح هناك مجرد مربع اختيار للمجموعة التي تريد رفعها وزر «(Upload)» [رفع]. ولم يعد هناك احتمال لإعادة فهرسة الموقع بأكمله أو لأخطاء أذونات الملفات الغريبة، لأن الأوامر الصحيحة أصبحت الآن مضمّنة في الشيفرة، وإن كان بإمكانك تغيير طريقة عمل تلك الأوامر في قائمة الإعدادات عند الحاجة.

إضافة إلى ذلك، أضفت شاشة مخرجات تُطلع المستخدم على ما يحدث في الوقت الفعلي. فإن حدث خطأ ما، تعرض رسالة خطأ وتسجلها في ملف سجل باستخدام مكتبة بایشون (loguru: Python (loguru) (GitHub - Delgan/loguru: Python (loguru) logging made stupidly simple). وقد جعل مسار العمل الناتج تصدير المجموعات من (ArchivesSpace) وتنظيفها ورفعها إلى موقع أدوات البحث أسهل بكثير وأوضح لأعضاء هيئة التدريس والموظفين لدينا، لكن كان هناك ما يمكن أن يجعله أيسر منالًا.

ومع أن النصوص البرمجية والواجهة الرسومية يباثون بسطت مسار العمل تبسيطًا هائلًا، فقد كنت بحاجة إلى طريقة لتوزيع الشيفرة على مستخدمينا بأسلوب مألوف لهم وبأكبر قدر ممكن من الوضوح. وقد ذكر توثيق (PySimpleGUI) إمكانية إنشاء ملف تنفيذي (.exe) لتطبيق الواجهة الرسومية. وإنشاء ملف تنفيذي من شأنه أن يحزم كل شيفرة التطبيق واعتمادياته ومفسر بايثون في ملف واحد. وهذا يتيح للمستخدم تشغيل التطبيق تمامًا كما يشغل أي تطبيق آخر يمكنه تنزيله وتثبيته على حاسوبه الشخصي (ويُسمى هذا أيضًا «التجميد» (freezing) ، يُنظر (Freezing Your Code)). ومن شأن ذلك أن يجعل مسار العمل أسهل لمستخدمينا. وسيسهّل أيضًا إيصال الشيفرة إليهم، لأنني لن أحتاج إلى تعليم مستخدمينا كيفية تحديث الشيفرة من (GitHub)، وهي منصة مشاركة الشيفرة لدينا. فأنشأت ملف (.exe) باستخدام (PyInstaller)، وقد أدى ذلك بالضبط: أنشأ ملفًا واحدًا أستطيع منه تشغيل تطبيقنا دون الحاجة إلى تثبيت بايثون، أو تحديث الشيفرة عبر (GitHub)، أو تشغيل أي أوامر في الطرفية.

ومضت خطوة أبعد، فاكشفت شيئًا يُسمى (Inno Setup)، وهو برنامج لإنشاء برنامج تثبيت ويندوز لتطبيقك. ومن شأن هذا أن يمكن مستخدمينا من تثبيت تطبيقنا التنفيذي تمامًا كما يثبتون أي برنامج آخر لويندوز، مع ميزة إضافية هي أن يعمل كأي تطبيق آخر على حواسيبهم. فهو يثبت التطبيق في ملف تعريف المستخدم المحلي، دون الحاجة إلى صلاحيات المسؤول لتثبيته أو تشغيله، ويثبت في مجلد خاص به، بحيث يمكن تخزين أي ملفات، مثل ملفات (EAD.xml) المصدرة أو ملفات السجل، في موقع مركزي افتراضي. ومع وجود ملف تنفيذي واحد وبرنامج تثبيت لويندوز معًا، أصبحت العملية مألوفة لمستخدمينا، لأنها تشبه تمامًا تثبيت أي نوع آخر من البرمجيات وتشغيله على حواسيبهم. ومع أنني حاولت إنشاء عملية مماثلة على حواسيب ماك، فلم أتمكن من إنجازها رغم وعود (PyInstaller) بالقدرة على إنشاء ملفات تنفيذية لويندوز ولينكس وماك. ورغم ذلك، شعرت بأنني قطعت شوطًا بعيدًا في تطوير البرمجيات؛ إذ أصبح لدينا تطبيق مكتبي مفتوح المصدر لويندوز، متكامل بواجهة رسومية وملف تنفيذي وبرنامج تثبيت. ثم ظهرت تعقيدات أخرى.

مشكلات مع ويندوز:

كانت أولى العقبات الكبرى تتعلق بويندوز نفسه، الذي يحاول حماية المستخدمين من تثبيت برمجيات قد تكون ضارة. فعند إنشاء برنامج تثبيت لويندوز دون شهادة تثبت هوية الناشر، يعرض النظام رسالة تحذير تنبه المستخدم إلى أن التطبيق صادر عن ناشر مجهول.

لا توضح رسالة التحذير مباشرةً كيفية متابعة تشغيل التطبيق. وللمضي قدمًا، يختار المستخدم «(More info)» [مزيد من المعلومات]، فيظهر زر «(Run anyway)» [التشغيل على أي حال] في أسفل النافذة.

كيف تتجنب هذا الخطأ أصلًا؟ يمكنك توقيع شيفرتك (code signing)، وهو ما يشير فعليًا إلى مايكروسوفت بأنك ناشر برمجيات مشروع، ويتيح لها تتبع أي مشكلات وصولًا إليك. وخلافًا لجميع الخطوات الأخرى المذكورة حتى الآن، وهي مجانية الاستخدام (باستثناء واحد ستحدث عنه لاحقًا)، يتطلب توقيع الشيفرة الدفع لمزود خدمة توقيع الشيفرة، مع مستويات مختلفة من الشهادات (يُنظر A guide to code signing certificates for the Microsoft (يُنظر reddit) على (r/electronjs) للاطلاع على تفصيل شامل لذلك). وإضافة إلى ذلك، ستحتاج إلى شراء وحدة أمان للأجهزة (Hardware Security Module, HSM) لتخزين الشهادة، إما محليًا أو في السحابة عبر مزود لهذه الوحدات. وفي هذه المرحلة، لم أكن مستعدًا لدفع مئات الدولارات لإزالة رسالة التحذير هذه، في حين كان أبسط حل هو إعلام مستخدمينا بكيفية تجاوزها.

كان هذا أول مؤشر على أن النشر الذاتي للبرمجيات ينطوي على مخاطره الخاصة، وأساسها مخاطر أمنية. ومن الممارسات الجيدة أن تضع مايكروسوفت هذه الضوابط، فكثير جدًا من البرمجيات التي تبدو مشروعة تأتي محزومة مع برمجيات خبيثة وفيروسات وكل ما يستطيع الفاعلون السيئون حشره في تطبيق مكتبي من مكونات ضارة. ونتيجة لرسالة التحذير هذه، يصبح المستخدمون أكثر حذرًا عند تنزيل التطبيقات التي يجدونها على الإنترنت وتشغيلها، حتى حين تكون تلك التطبيقات مطورة داخليًا لمسار عمل متخصص جدًا. وفي إصداراتنا القليلة الأولى، علّمت أعضاء هيئة التدريس والموظفين لدينا كيفية تجاوز هذه الرسالة، مؤكدًا لهم أن لا شيء مما برمجته ينبغي أن يضر بحواسيبهم. غير أن برامج فحص الفيروسات لم توافق على ذلك.

مشكلات مع برامج فحص الفيروسات:

مع أن عمليتنا لإنشاء ملفات (.exe) وبرامج تثبيت ويندوز بدت ناجحة إلى حد كبير، فإننا لم نتوقع أن تلتقط برمجيات فحص الفيروسات تطبيق بايثون المطور داخليًا وتصفه على أنه ضار. فقد أبلغني أحد المستخدمين أنه يواجه صعوبة في تشغيل التطبيق من حاسوبه؛ إذ حدّدت مايكروسوفت التطبيق على أنه فيروس، وعزلت برنامج التثبيت وملفات التطبيق قسرًا وأزالته. ولأنني لم أفهم سبب ذلك، تواصلت مع مسؤول الأنظمة لدينا في قسم تقنية المعلومات طلبًا للمساعدة. فاقترح تمرير ملف التثبيت عبر (VirusTotal)، وهي أداة فحص على الإنترنت تستخدم كثيرًا من برامج مكافحة الفيروسات لفحص ملف أو عنوان (URL) لمعرفة ما إذا كان الملف مصنفًا على أنه خبيث (How it works VirusTotal). وعند رفع برنامج التثبيت والملف التنفيذي، اكتشفنا أن مايكروسوفت وعدًا كبيرًا من برامج مكافحة الفيروسات الأخرى صنّفت التطبيق على أنه خبيث. وهذا ما يُعرف بـ«الإشارة الكاذبة» (false-positive).

الإشارة الكاذبة هو أن يصنّف برنامج لمكافحة الفيروسات برنامجًا آخر على أنه فيروس، رغم أنه ليس كذلك. ويمكن أن يحدث هذا حين يستخدم فاعلون خبيثون برنامجًا مفتوح المصدر لتوزيع برمجيات خبيثة على المستخدمين (يُنظر (What's a False Positive and How Can You Fix It? | All About Cookies) لمزيد من التفاصيل). وأدركنا، أنا ومسؤول الأنظمة، أن (PyInstaller) هو أصل المشكلة؛ إذ كان يُصنّف فيروسًا لأن آخرين استخدموه لنشر شيفرة خبيثة (False-positive search - PyInstaller Google Group). فحاولنا أولًا تجميع (PyInstaller) محليًا بدلًا من سحبه من شيفرة (PyInstaller) على الإنترنت، مستعينين بهذا الدليل: (Pyinstaller EXE False-Positive Trojan Virus [RESOLVED]). ومع أن ذلك منع ظهور تقارير الإنذارات الكاذبة في المرات القليلة الأولى، فسرعان ما بدأت برامج أخرى لمكافحة الفيروسات تصفّه، مع مزيد من الإنذارات الكاذبة. فقررنا تجربة حظنا مع حزمة أخرى لتوليد الملفات التنفيذية، واستقر اختيارنا على (cx-Freeze) (GitHub - marcelotduarte/cx_Freeze). ومع أنها تطلبت إعدادًا أكثر من (PyInstaller)، فقد أظهرت اختباراتنا الأولية نتائج واعدة، إذ لم يصنّفها أي برنامج لمكافحة الفيروسات على أنها فيروس.

وإضافة إلى ذلك، عملنا معًا على أتمتة عملية توليد الملف التنفيذي وحزمه مع برنامج تثبيت ويندوز بأكملها باستخدام (GitHub Actions)، مما جعل عملية التوزيع على مستخدمينا أسهل وأسرع بكثير (GitHub Actions documentation). ولسوء الحظ، لم يمض وقت طويل حتى عادت الإنذارات الكاذبة، فلجأنا إلى آخر وسيلة متاحة لنا: الإبلاغ عن الإنذارات الكاذبة. فإذا صنّف برنامج لمكافحة الفيروسات ملفًا على أنه خبيث وأنت تعلم أنه ليس كذلك، يمكنك تقديم بلاغ إلى الجهة المطورة لذلك البرنامج تذكر فيه أن الملف صنّف خطأً (إنذار كاذب). ولدى (VirusTotal) قائمة بجميع وسائل الاتصال ببرامج مكافحة الفيروسات التي يستخدمها، بحيث يمكنك الإبلاغ عن هذه الإنذارات الكاذبة. وبعد تقديم بلاغ عن إنذار كاذب إلى أحد مزودي برامج مكافحة الفيروسات، استجاب في غضون

أيام قليلة وأزال تصنيفه. غير أن برنامجًا آخر لمكافحة الفيروسات لم يرد قط على بلاغنا، ولا يزال تصنيفه يظهر في نتائج (VirusTotal). ومع أن بعض مزودي البرمجيات أكثر استجابة من غيرهم، فقد توصلنا على الأقل إلى طريقة للتصدي للإنذارات الكاذبة، وأصبحنا الآن نرفق تقريرًا من (VirusTotal) مع جميع أحدث إصداراتنا، حريصين على التنبيه إلى الإنذارات الكاذبة حين تظهر والإبلاغ عنها بوصفها كذلك إلى برامج فحص الفيروسات المعنية. ونأمل أن تبني هذه الخطوات الصغيرة الثقة مع مستخدميها. فنحن نريد أن نُظهر أننا نبذل قصارى جهدنا لضمان أن البرمجيات التي نوزعها مفحوصة وآمنة الاستخدام، إلى جانب كونها واضحة وسهلة.

مشكلات مع التراخيص:

بعد أن أصبحت لدينا عملية لإدارة الإنذارات الكاذبة، بقيت مسألة أخيرة كان علينا تجاوزها، وتتعلق بتغييرات في نموذج الترخيص الخاص بمكتبة الواجهة الرسومية لبايثون التي كنا نستخدمها. فبدلاً من عام 2024، أعلنت (PySimpleGUI) أنها تغيّر نموذج ترخيصها، فتصنّف المستخدمين إلى «هواة» (Hobbyist) و«تجاربيين» (Commercial). ووفقاً لتعريف (PySimpleGUI) نفسها، فإن «المطورين في المؤسسات التعليمية الذين يستخدمون (PySimpleGUI) لأغراض إدارية أو بحثية يُعدّون مطورين تجاريين، ويسجلون بوصفهم مطورين تجاريين، ويدفعون رسوم الترخيص المقابلة البالغة 99 دولارًا». (FAQ - PySimpleGUI Documentation). ويحتاج المستخدمون التجاريون إلى دفع 99 دولارًا عن كل مطور للحفاظ على مفاتيح المطور والتوزيع الخاصة بهم، مقابل ترخيص دائم للإصدارات المدعومة خلال عام واحد من الشراء الأول. وكان من الأسباب الرئيسة لهذا التغيير جعل التطوير المستمر للمكتبة ممكناً بمصدر دخل أكثر انتظاماً، وضمان أصالة إصدار (PySimpleGUI) الذي تستخدمه. وهذا يخلق تعقيدات إضافية لأننا أصدرنا التطبيق بترخيص (CC-BY-SA-4.0)، الذي يسمح لأي شخص باستخدام الشيفرة أو تنزيل الملفات القابلة للتوزيع. وهذا قد يتعارض مع ترخيص (PySimpleGUI) نفسه، لذا لسنا متأكدين تماماً من كيفية التعامل مع ذلك مستقبلاً. ولحسن الحظ، تظل الإصدارات السابقة من تطبيقنا متاحة على الإنترنت، وستظل قابلة للتنزيل والاستخدام، وإن كان تحديث (PySimpleGUI) إلى الإصدار 5 أو ما بعده سيتطلب منا شراء ترخيص وتضمين مفتاح توزيع في شيفرة التطبيق للتحقق من مشروعته. وفي الوقت الحالي، قررنا الدفع مقابل ترخيص لمدة عام واحد لنمنح أنفسنا وقتاً للتفكير في واجهات رسومية بديلة ((tkinter) أو (FreeSimpleGUI) هما أكثر بدائلنا قابلية للتطبيق) وفي الطريقة التي سنوزع بها مسار العمل مستقبلاً. إن فهم مشهد تراخيص البرمجيات يتجاوز تدريجي بوصفي أرسيفيًا، لكنه يكشف طبقة أخرى من التعقيد ينبغي مراعاتها عند بناء تطبيقاتك المكتوبة الخاصة ومشاركتها.

الخاتمة:

جاء التعمق في تطوير البرمجيات مصحوباً بتحديات غير متوقعة تتعلق ببناء مسار عملنا وتوزيعه وصيانتته. وقد علّمني ذلك قيمة جعل هذا النوع من التطبيقات المكتوبة أسهل استخداماً لمستخدمينا ولي أنا كلما أمكن. فإ إنشاء واجهة رسومية لتصدير ملفات (EAD.xml) من (ArchivesSpace)، وتنظيف تلك الملفات، ورفعها إلى موقعنا جعل العملية أسهل بكثير للمستخدمين، لكنه أدى إلى نتيجة غير مقصودة هي إعادة النظر في كيفية ترخيص تطبيقنا مستقبلاً. وتوليد ملفات تنفيذية تجعل تشغيل مسار العمل أمرًا يسيرًا، دون الحاجة إلى تثبيت بايثون أو أي من اعتمادياتها، بسّط استخدام الشيفرة ومشاركتها، لكنه أجبرنا أيضًا على إيجاد طريقة للتصدي لتقارير الإنذارات الكاذبة من برامج

فحص الفيروسات، وصنع برنامج تثبيت لويندوز جعل تثبيت التطبيق واضحًا ومألوفًا، وتغليف العملية كلها بـGitHub Actions) جعل توليد الإصدارات الجديدة عملية أكثر سلاسة. كما يبين ذلك أن بناء الثقة مع المستخدمين يتطلب جهدًا مقصودًا، وأن بناء الثقة مع مايكروسوفت يتطلب مألًا (إن كنت مستعدًا لدفعه).

إن إدراك مزالقي بناء برمجياتك الخاصة للمستخدمين أظهر لي بجلاء مدى تعقيد هذه العملية وكثرة وسائل الحماية المحيطة بها. فهناك كثير من الأشخاص والعمليات التي تحمي المستخدمين بنشاط من الشيفرات الخبيثة والفاعلين السيئين، وهذا يستحق العقبات الإضافية التي علينا اجتيازها إن كنا نصنع برمجياتنا الخاصة بنيات حسنة. ولدي تحفظاتي الخاصة على الدفع مقابل أشياء مثل توقيع الشيفرة، لكنني أفهم أن المقصود هو جعل منتج الشيفرة مسؤولًا أمام المستخدمين. وتوجد أدوات مثل (VirusTotal) لمساعدة المطورين، حتى من وجدوا أنفسهم مطورين دون تخطيط مسبق، على فهم كيفية تقييم الأدوات التي نستخدمها من حيث احتمال استخدامها لأغراض خبيثة. لقد تعمقت في تطوير التطبيقات، وواجهت تعقيدات أمن التطبيقات وتوزيعها وترخيصها، مدفوعًا برغبة في جعل عمل زملائي الأرشيفيين أسهل قليلًا. ومع أنني أرى أن الأمر يستحق الجهد للمساعدة في إنشاء مسارات عمل سهلة الاستخدام لزملائنا، فإن كنت تنوي خوض التجربة نفسها، فانتبه جيدًا إلى خطواتك.

ملحق:

استلهمت كتابة هذا المقال من مؤتمر (Code4Lib) في العام الماضي، حيث تحدث أحد العروض عن إنشاء واجهة رسومية ببايثون واستخدام (PyInstaller) لأحد مسارات العمل الداخلية لديهم. وسألتهم إن كانوا يخططون لتوزيعه أو إتاحتها للعموم، لكنهم قالوا إنهم سيبقونه للاستخدام الداخلي فقط. وسمعت أشخاصًا آخرين يتحدثون عن صنع أشياء مماثلة (ولا سيما حين غيّرت (PySimpleGUI) نموذج ترخيصها) في قناة (#python) على (Slack) الخاصة بـ(Code4Lib)، لذا كنت آمل أن يسلط هذا المقال الضوء على بعض المشكلات التي يمكن للناس أن يتبهاوا إليها إن قرروا الذهاب إلى المدى الذي ذهبت إليه في صنع تطبيقات مكتوبة مولدة ببايثون.