Python (tkinter/pyinstaller) 製 Windows 実行ファイルの配布方法比較分析

1. はじめに

本レポートは、Python (tkinter/pyinstaller) を用いて作成された複数の小規模な Windows アプリケーション（実行ファイル形式）を、エンドユーザーにとって負担の少ない形で配布・共有する方法を求める開発者の課題に対応するものです。現在ソースコードは GitHub で共有されているものの、利用者が実行環境を構築する必要があり、より手軽な利用方法が模索されています。特に、ファイルサイズが大きくなりがちな実行モジュールを多数登録する際の GitHub の適性への懸念、および Windows ストアアプリとしての公開可能性が焦点となっています。

この課題認識に基づき、本レポートでは GitHub Releases 機能の適切な利用法、代替となるソフトウェア配布プラットフォーム、汎用クラウドストレージの活用、そして Microsoft Store での公開手順と要件について、技術的側面、コスト、管理の容易さ、エンドユーザーの利便性といった多角的な視点から分析・比較し、最適な配布戦略の選択に資する情報を提供します。開発者が直面する、ソースコード共有から実行可能バイナリ配布への移行という一般的な過程を背景に、具体的なツールセット（Python, tkinter, pyinstaller）とターゲットプラットフォーム（Windows）に即した検討を行います。

2. PyInstaller アプリケーションの理解

配布戦略を検討する前に、配布対象となる PyInstaller 製アプリケーションの特性を理解することが重要です。

2.1. PyInstaller の概要

PyInstaller は、Python アプリケーションとその依存関係（ライブラリ、モジュール等）を一つのパッケージにまとめるツールです [1]。これにより生成されたパッケージは、実行環境に Python インタープリタや特定のモジュールがインストールされていなくても動作します [1]。これは、エンドユーザーに Python 環境のセットアップを要求するという、開発者が直面している根本的な問題を解決します [2]。PyInstaller は、必要な依存関係を自動的に検出し、Python インタープリタ自体も含めてバンドルするため、ユーザーは Python プロジェクトであることを意識せずにアプリケーションを実行できます [2]。

2.2. 出力形式とそのトレードオフ

PyInstaller は主に二つの出力形式を提供します。

- **One-Folder (--onedir)**: これがデフォルトのモードです。実行ファイルと全ての依存ファイルを含むフォルダが生成されます [4]。開発者にとっては、フォルダ内のファイル構成を確認できるため、ビルド時の問題（依存関係の不足など）をデバッグしやすい利点がありま

す[2]。一方で、ユーザーから見ると複数のファイルが存在するため、単一ファイルよりも直感的でない可能性があります。
- **One-File (--onefile)**: 単一の実行可能ファイル（.exe）を生成します[4]。エンドユーザーにとっては「このファイルを実行するだけ」というシンプルな配布形態となり、理解しやすい利点があります[4]。ただし、実行時に内部のファイルを一時フォルダに展開するため、One-Folder モードに比べて起動がわずかに遅くなることがあります[4]。また、依存関係の問題が発生した場合のデバッグは、単一ファイルに全てが隠蔽されているため難しくなる傾向があります[2]。

開発プロセスにおいては、まず One-Folder モードでビルドとテストを行い、問題がないことを確認した上で、配布用に One-File モードに切り替えるという手順が推奨されます[4]。これにより、デバッグの容易さと配布の簡便さのバランスを取ることができます。

2.3. プラットフォーム固有性

重要な点として、PyInstaller で生成された実行ファイルは、それを作成したオペレーティングシステム（OS）でのみ動作します[6]。つまり、Windows 上でビルドされた.exe ファイルは Windows でしか実行できず、macOS や Linux では動作しません[6]。これは、開発者が「Windows 限定」で配布したいという要件には合致していますが、将来的にクロスプラットフォーム対応を検討する場合には制約となります。

2.4. まとめ

PyInstaller は、Python アプリケーションを自己完結型の Windows 実行ファイルにパッケージ化する問題を解決します[3]。One-Folder と One-File のどちらの形式を選択するかは、開発時のデバッグのしやすさと、エンドユーザーへの配布時の分かりやすさのトレードオフとなります。生成される実行ファイルは Windows 専用である点も認識しておく必要があります。次のセクションでは、こうして作成された実行ファイルをどのように配布するか、具体的な選択肢を分析します。

3. 配布オプション分析

PyInstaller で実行ファイルを作成した後、それをユーザーに届けるための様々な方法が存在します。それぞれの特徴、利点、欠点を比較検討します。

3.1. オプション1: GitHub Releases

開発者は現在ソースコード管理に GitHub を利用しており、実行ファイルの配布にも GitHub を検討していますが、ファイルサイズの懸念から不適と考えています。しかし、この認識は GitHub の機能を正確に捉えていない可能性があります。

- メカニズム: GitHub Releases は、コンパイル済みのソフトウェア、リリースノート、バイナリファイルへのリンクなどを、ソースコードの特定のバージョン（タグ）に関連付けて配布す

るために設計された機能です [7]。これは、Git リポジトリ内で直接大きなファイルをバージョン管理するのとは異なります。

- ファイルサイズ制限の誤解: 開発者の懸念とは異なり、GitHub Releases は個々のファイルサイズの上限が **2 GiB** と非常に大きく設定されています [7]。さらに、リリース全体の合計サイズやダウンロードに使われる帯域幅には制限がありません [7]。これは、典型的な PyInstaller 製の実行ファイル（たとえ多数の小さなツールであっても）を配布するには十分すぎる容量です。
- リポジトリ制限との違い: GitHub が警告を発する 50 MiB 超過や、ブロックする 100 MiB 超過といった厳しい制限は、あくまで Git のコミット履歴に直接含まれるファイルに対して適用されるものです [7]。GitHub は、大きなバイナリファイルを配布する際には、Git で追跡する代わりに Releases 機能を使用することを明確に推奨しています [7]。
- **Git LFS** との関連性の低さ: Git Large File Storage (LFS) は、音声サンプル、動画、データセットなどの大きなファイルを Git リポジトリ内で効率的にバージョン管理するための技術であり、実ファイルへのポインタをリポジトリに格納します [10]。LFS にもプランに応じたファイルサイズ制限がありますが [10]、これはリポジトリ内のアセット管理を目的としており、最終的なリリースバイナリを配布するための機能ではありません。その目的には Releases が適しています [7]。LFS について言及するのは、Releases 機能との違いを明確にし、開発者のユースケースには Releases が適切であることを強調するためです [14]。
- 多数のツールの管理: Releases は Git のタグに関連付けられます。多数のツールを配布する場合、以下の戦略が考えられます。
  - ツールごとに個別のリポジトリを作成し、それぞれで Releases を管理する。
  - 全てのツールを単一のリポジトリ（例: サブディレクトリ構成）に含め、複数のツールの実行ファイルをまとめて一つのリリースとするか、特定のツール向けのリリースであることを示すタグ（例: toolA-v1.0, toolB-v1.2）を使用する。後者の場合、ツールが非常に独立して更新されるなら、リリース管理がやや煩雑になる可能性があります。
- 利点: 無料で利用可能、寛大な制限、ソースコードとの連携、開発者にとって馴染み深いプラットフォーム、十分な帯域幅。
- 欠点: リポジトリ訪問者以外への発見性は低い、単一リポジトリで多数のツールを管理するには工夫が必要、Windows から「信頼されていないアプリケーション」として警告が表示される可能性がある。

GitHub Releases に関する重要な点は、開発者の当初の懸念（ファイルサイズ制限）が、リポジトリ内のファイル追跡と Releases 機能の混同に基づいている可能性が高いということです。Releases 機能の 2 GiB/ファイルという制限と無制限の帯域幅は [7]、開発者のニーズに十分応えられます。より厳しい制限 [7] や Git LFS [10] は、この配布シナリオには直接関係ありません。したがって、GitHub Releases は技術的には非常に有力な選択肢です。ただし、多数の独立した小ツールを効果的に管理するためには、リポジトリ構成（個別リポジトリか、タグ付け戦略を持つ単一リポジトリか）を意識的に決定する必要があります。ツールが独立して進化する場合、単一リポジトリでのリリース管理は煩雑になる可能性があるため、構造的な選択が求め

られます。

3.2. オプション2: 専用ソフトウェアプラットフォーム (itch.io, SourceForge)

GitHub 以外にも、ソフトウェア配布に特化したプラットフォームが存在します。

- **itch.io**:
  - 概要: 主にインディーゲーム開発者に人気がありますが、ツールやソフトウェアの配布にも適したプラットフォームです [16]。開発者にとっての配布の容易さと、ユーザーにとってのシンプルなダウンロード体験に重点を置いています。
  - 配布プロセス: 単純なアーカイブファイル（.zip, .7z など）をアップロードするか、コマンドラインツール butler を使用することを推奨しています [16]。butler を使うと差分アップデートが可能になり、ユーザーは変更点のみをダウンロードすれば済むため、アーカイブ全体を再ダウンロードするよりも効率的です [18]。起動設定や前提条件を指定するためのマニフェストファイルもサポートされています [16]。プラットフォーム（Windows, macOS, Linux）をタグ付けすることで、ユーザーが適切なバージョンを見つけやすくなります [18]。
  - ユーザーエクスペリエンス: 特に itch.io のデスクトップクライアント経由でダウンロードした場合、Windows による「信頼されていない実行ファイル」の警告を軽減する効果が期待できます [17]。プロジェクトごとに専用のページが提供されます。
  - コスト: 無料ソフトウェアのアップロードと配布は無料です。有料ソフトウェアの場合は、設定可能な収益分配率に基づいた手数料がかかります。
  - 利点: 開発者向けツール (butler) が便利、効率的なアップデート機能、プラットフォーム内での発見可能性、信頼性向上や警告軽減の可能性、無料ツールは無料。
  - 欠点: 主にゲームで知られているため、汎用ツールの発見性は専門的なソフトウェアポータルより低い可能性がある。

  itch.io は単なるファイルホスティングを超えた価値を提供します。butler による効率的なパッチベースのアップデート [18] や、クライアント利用による警告の軽減可能性 [17] は、基本的なファイルリンクを提供するだけの方法と比較して、ユーザーエクスペリエンスを直接的に向上させます。これは、配布プロセスそのものに付加価値を与えるプラットフォームであり、ユーザーの「手軽に使いたい」という目標によく合致しています。
- **SourceForge**:
  - 概要: 主にオープンソースプロジェクト向けに長年利用されているプラットフォームで、ファイルホスティングとプロジェクト管理ツールを提供しています [19]。膨大な数のプロジェクトと実行可能ファイルがホストされています [19]。
  - 配布プロセス: 基本的にはファイルリポジトリとして機能します。開発者はプロジェクトのファイルセクションに実行ファイルやアーカイブをアップロードします [22]。ユーザーはそれを閲覧し、直接ダウンロードします。
  - ユーザーエクスペリエンス: 多くのユーザーにとって馴染みのあるダウンロードインターフェースですが、ダウンロード時に広告やバンドルソフトの提案が表示されることがあり、体験を損なう可能性があります。

    - コスト: オープンソースプロジェクトは無料です。
    - 利点: 定評のあるプラットフォーム、大容量ダウンロードに対応、オープンソースなら無料。
    - 欠点: インターフェースが古く感じられることがある、ダウンロード体験に広告などが含まれる場合がある、itch.io や GitHub Releases のような洗練されたプレゼンテーションやアップデート機構には重点が置かれていない。主にオープンソースに関連付けられている（これはユーザーの現状と一致しますが、バイナリ配布の必須条件ではないかもしれません）。

SourceForge は、GitHub のソースコード統合型リリースモデル [7] や itch.io のインディー向けツール群 (butler など) [16] と比較して、より伝統的なダウンロードリポジトリのモデル [19] を代表します。ユーザーの実行ファイルをホストする機能は十分に果たしますが、アップデート管理やユーザーの信頼性向上といった付加価値機能は他の専門プラットフォームに比べて少ないです。その強みは、特にオープンソースソフトウェアにおける、長年の実績と大規模なファイルミラーとしての能力にあります。

3.3. オプション3: Microsoft Store

Windows の公式アプリケーションストアであり、広範なリーチ、ユーザーからの信頼、自動アップデートといった大きな利点を提供します。

- **Python/PyInstaller** アプリへの適合性: Microsoft Store はプログラミング言語を区別しません。適切にパッケージ化されていれば、Python アプリケーションも提出可能です [23]。重要なのは言語ではなく、パッケージング方法です。
- アカウント設定と費用: Microsoft Partner Center を通じて開発者アカウントの登録が必要です [24]。登録には一回限りの料金がかかり、個人アカウントは約 19 USD、会社アカウントは約 99 USD です（正確な金額は国/地域によって異なります）[24]。日本では個人アカウントが 1,847 JPY、会社アカウントが 9,800 JPY です [27]。
- パッケージング: MSIX が推奨される最新の形式で、クリーンなインストール/アンインストール体験を提供します [24]。しかし、従来の .exe や .msi インストーラーも受け付けられます [25]。PyInstaller の出力をシンプルなインストーラーツールでラップする場合、こちらの方が初期段階では容易かもしれません。パッケージングには、ロゴやアイコンといったアセットの準備と、マニフェストファイルの作成が必要です [24]。Visual Studio は MSIX パッケージング用のツールを提供しますが [24]、他の方法も存在します [23]。
- 提出プロセス: Partner Center でアプリのリスト情報（名前、説明、スクリーンショット、価格設定、提供地域、年齢区分、プライバシーポリシーなど）を設定し、パッケージをアップロードします [24]。Windows App Certification Kit (WACK) のテストに合格し [24]、Microsoft によるコンプライアンス、セキュリティ、コンテンツ適合性の審査を通過する必要があります [24]。このプロセスには通常数日かかります [24]。API を利用した自動化も可能ですが [28]、複雑さが増します。
- 多数のツールの管理: 各ツールはおそらく個別のアプリとして提出する必要があります。これは、「多数の」小さなツールに対して、リスト設定やパッケージング、提出といったプロ

セスを繰り返すことを意味し、大きなオーバーヘッドになる可能性があります。
- 潜在的なコスト要因: 古い情報源 [30] によれば、過去には無料アプリの無料提出回数に制限があり（アップデートや認証失敗もカウントされる）、それを超えると提出ごとに料金が発生したケースがありました。これは古いポリシーや Windows Phone 固有のものかもしれませんが、多数の無料ツールを頻繁に提出する場合、潜在的なコスト要因として Partner Center の最新ドキュメントで確認する価値があります。
- 価格設定/試用版: アプリを無料にするか、有料にするか、また期間限定の無料試用版を提供するかを選択できます [25]。
- 利点: ユーザーからの高い信頼性、Windows 内での発見可能性、自動アップデート、合理化されたインストール/アンインストール（特に MSIX）。
- 欠点: 初期費用（登録料）、アプリごとの大きなオーバーヘッド（パッケージング、提出、審査）、迅速なアップデート展開には不向きな場合がある、他の方法に比べて直接的なコントロールが少ない。多数の非常に小さな、頻繁に更新されるツールを管理するには、このオーバーヘッドは非現実的かもしれません。

Microsoft Store は、ユーザーの信頼と OS との統合という点で最高レベルの正当性を提供しますが、アプリケーションごとに最も大きなオーバーヘッド（コスト、プロセス、パッケージング）を課します。ユーザーは一般的に、ダウンロードした EXE よりも公式ストアのアプリを信頼し、ストアはアップデートを自動的に処理します。しかし、プロセスには料金 [24]、特定のパッケージング要件 [24]、リスト設定 [25]、審査サイクル [24] が伴います。一つか二つの主要なアプリケーションであれば価値があるかもしれませんが、「多数の小さなツール」に対してこのプロセスを繰り返すことは、時間的、そして潜在的には金銭的な（無料アプリの提出制限が存在する場合 [30]）大きな投資となります。これは明確なトレードオフを生み出します：正当性とリーチを得る代わりに、ツールごとに多大な管理努力が必要になります。

また、MSIX を使うか EXE/MSI を使うかに関わらず、ストアへの提出には、基本的な PyInstaller の出力 [4] を超える、正式なパッケージング手順が必要です。ストアは、できれば MSIX [24]、少なくとも準拠した EXE/MSI インストーラー [25] を要求します。これは、pyinstaller を実行した後、ストアに提出する前に、Visual Studio のパッケージャー [24] やインストーラー作成ツール（後述）などを使用する追加のステップが必要であることを意味します。これは、PyInstaller の出力を直接 GitHub Releases や itch.io にアップロードするのに比べて、複雑さを増します。

3.4. オプション4: 汎用クラウドストレージ (Google Drive, Dropbox, OneDrive など)

- 方法: 実行ファイル（.exe または One-Folder 出力を含む.zip ファイル）をクラウドストレージサービスにアップロードし、公開ダウンロードリンクを共有します。必要であれば、シンプルなウェブサイトを作成するか、GitHub リポジトリの README ファイルにツールとそのダウンロードリンクのリストを掲載します。
- 利点: アップロードプロセスが非常に簡単、無料プランでも小規模ツールには十分なスト

レージ容量が提供されることが多い、ユーザーがこれらのサービスに慣れている。
- 欠点: バージョン管理やアップデート機構が組み込まれていない（ユーザーは手動で再ダウンロードが必要）、設定によってはダウンロードリンクが扱いにくかったりサインインが必要になったりすることがある、専用プラットフォームに比べてプロフェッショナルさに欠ける、ファイル移動や削除でリンク切れが発生する可能性がある、Windows から「信頼されていないアプリケーション」警告が表示される可能性が高い、発見可能性を高める機能がない。無料プランの帯域幅制限は、ダウンロード数が多い場合に問題になる可能性がある。

クラウドストレージは最も簡単なアップロード方法ですが、ソフトウェア配布に必要な機能（バージョン管理、アップデート通知、発見可能性など）はほとんど提供されません。GitHub Releases (タグによるバージョン追跡)、itch.io (butler アップデート)、Microsoft Store (自動アップデート、発見可能性) のようなソフトウェア配布プラットフォーム固有の機能が欠けています。アップデートの発見やバージョン管理の負担は完全にエンドユーザーにかかります。非常に非公式な共有には使えますが、多数のツールを管理したり、洗練されたユーザーエクスペリエンスを提供したりするには、管理上も機能上もスケールしません。

3.5. オプション**5:** セルフホスティング

- 方法: 個人用またはレンタルしたウェブサーバー（VPS、十分な権限のある共有ホスティングなど）をセットアップし、実行ファイルを直接ホストします。ダウンロード用のリンクを含むウェブページを作成します。
- 技術要件: ウェブサーバー管理（Apache, Nginx など）の知識、ドメイン名登録、セキュリティ管理（HTTPS 証明書）、十分な帯域幅とストレージの確保が必要です。
- 利点: プレゼンテーション、配布、ユーザーエクスペリエンスを完全にコントロールできる。プラットフォームの制約や手数料がない（ホスティング費用を除く）。必要であればカスタムのアップデート機構を実装できる（ただし複雑）。
- 欠点: 技術的な手間と継続的なメンテナンス責任が最も大きい。ホスティングとドメイン名に関連するコストが発生する。帯域幅の使用量に責任を持つ必要がある。発見可能性は完全に外部プロモーションに依存する。Windows から「信頼されていないアプリケーション」警告が表示される可能性が高い。

GitHub、itch.io、Microsoft Store のようなプラットフォームは、ファイルホスティング、帯域幅管理、ダウンロードインターフェース提供といったインフラストラクチャを抽象化してくれます。セルフホスティングでは、開発者がこれら全ての側面を直接管理する必要があります。これは究極の柔軟性を提供しますが、技術スキル、セットアップとメンテナンスの時間、そして直接的なホスティングコストを要求します。多数の小さなツール作成に集中したい開発者にとって、配布インフラ自体の管理オーバーヘッドは、開発時間を大幅に奪う可能性があります。つまり、セルフホスティングは最大のコントロールを可能にする代わりに、最大の努力と責任を伴います。

4. ユーザーエクスペリエンスの向上: インストーラー

PyInstaller の出力をそのまま配布する代わりに、インストーラー作成ツール（例: Inno Setup, NSIS など、これらは例であり、提供された情報には含まれていません）を使用して、実行ファイル（単一.exe または One-Folder の内容）をラップすることを検討できます。

- 利点: Windows ユーザーにとって馴染みのあるインストール体験を提供します。デスクトップやスタートメニューへのショートカット作成、ファイルタイプの関連付け、アンインストーラーの提供、前提条件の管理（PyInstaller がほとんどをバンドルすることを目指していますが）などを処理できます。アプリケーションをより「プロフェッショナル」に見せることができます。特定の配布方法（例: Microsoft Store に.exe インストーラーとして提出する場合 [25]）では必要になる場合があります。
- 欠点: ビルド/リリースプロセスにさらに一手間加わります。

インストーラーの使用は、開発者の「手軽につかえるようにしたい」という目標に貢献します。特に技術に詳しくないユーザーにとっては、単純な.exe ファイルよりもインストーラーの方が安心感があり、使いやすいと感じられる可能性があります。

5. 比較分析

これまでの分析結果を統合し、開発者の主要な基準に基づいて各配布オプションを明確に比較します。

機能	**GitHub Releases**	**itch.io**	**SourceForge**	**Microsoft Store**	クラウドストレージ **(例: Drive)**	セルフホスティング
アップロード容易性 **(開発者)**	中 (Gitタグ + アップロード)	易 (Web UI or butler CLI)	易 (Web UI アップロード)	難 (パッケージング + 提出)	極易 (ドラッグ&ドロップ)	中～難 (サーバー設定)
多数ツールの管理	リポジトリ戦略が必要	良 (プロジェクト毎ページ)	中 (プロジェクトファイルリスト)	高負荷 (アプリ毎提出)	手動 (リンク整理)	手動 (ファイル/ページ整理)
ストレージ制限	2GiB/ファイル, 合計制限なし	寛大 (詳細は変動)	寛大 (詳細は変動)	ストアが処理	段階的 (無料枠あり)	ホスティングプラン依存
帯域幅/コスト	無料	無料 (無料アプリの場合)	無料 (OSプロジェクトの場合)	開発者登録料 (1回 $19/$99)	段階的 (無料枠あり)	ホスティング費用発生

アップデート機構	手動DL (新リリース)	butler (差分)/手動DL	手動DL	自動	手動DL	手動 / カスタム
エンドユーザー体験	簡単DL, 警告の可能性	良 (クライアント補助), 更新容易	簡単DL, 広告の可能性	シームレスなInstall, 信頼性高	簡単DL, 警告の可能性	簡単DL, 警告の可能性
発見可能性	低 (リポジトリ限定)	中 (プラットフォーム内検索)	中 (プラットフォーム内検索)	高 (ストア検索/ブラウズ)	無	無 (プロモーション必要)
信頼性 (警告)	警告の可能性高い	クライアント経由で低減可能性	警告の可能性高い	高 (ストア検証済)	警告の可能性高い	警告の可能性高い
MS Store 特有事項	N/A	N/A	N/A	登録料, パッケージング, 審査必須	N/A	N/A

この比較表は、開発者が複数の選択肢にわたる複数の要因を比較検討する上で役立ちます。各行（機能）または各列（オプション）を見ることで、特定のニーズ（例: 最低コスト、簡単なアップデート、最高の信頼性）に対してどのソリューションがどのように適合するかを素早く把握できます。これにより、異なる配布方法間の主要なトレードオフが明確になり、情報に基づいた意思決定が促進されます。

例えば、GitHub Releases と itch.io を比較すると、既存プラットフォームの活用とソース連携の利便性 [7] 対、より優れたアップデート機構 (butler) と信頼性向上（警告軽減）の可能性 [17] というトレードオフが見えます。また、専用プラットフォーム（GitHub, itch.io）と Microsoft Store を比較すると、導入の容易さや低コスト [7] 対、広範なリーチ、最高の信頼性、そしてそれに伴う高いオーバーヘッド [24] という対立軸が明らかになります。シンプルなホスティング（クラウドストレージ、GitHub）と管理されたプラットフォーム（itch.io, Store）の間では、最小限の努力 [7] 対、付加価値機能（アップデート、発見可能性など） [18] の選択となります。

6. 推奨事項

分析結果に基づき、開発者の優先順位に応じた推奨事項を以下に示します。

- 最も簡単な開始 / 低フリクション: **GitHub Releases**
  - 既存の GitHub 利用を活用できます。ファイルサイズ制限に関する誤解 [7] を解消し、この機能がニーズに適していることを認識することが第一歩です。多数のツールを管理するために、明確なリポジトリ戦略（例: まずはツールごとにリポジトリを作成）を立

てることを推奨します。
- ストアのオーバーヘッドなしで最良のユーザー体験 (アップデート/信頼性): **itch.io**
  - butler による効率的なアップデート [18] と、クライアント経由での警告軽減の可能性 [17] が大きな利点です。インディー開発者の雰囲気にも合っています。
- 最大のリーチ / プロフェッショナルさ (労力/コスト投資を厭わない場合): **Microsoft Store**
  - 公式なプレゼンスを望み、関連するオーバーヘッド [24] を受け入れる準備がある場合に適しています。全てのツールを一度に登録するのではなく、まず一つか二つの主要なツールで試してみることを推奨します。パッケージング [24] と一回限りの登録料 [26] が必要であることを念頭に置くべきです。
- 絶対的に最も簡単なアップロード (最小限の機能): 汎用クラウドストレージ
  - バージョン管理、アップデート、プレゼンテーションが重要でない場合にのみ検討すべき選択肢です。
- 「多数の小さなツール」への対応: どのプラットフォームを使用するにしても、多数のツールを管理するにはオーバーヘッドが伴います。しかし、Microsoft Store [24] はツールごとのオーバーヘッドが最も高くなる可能性が高い一方、GitHub Releases [7] や itch.io [18] は、適切に構成すればよりスケーラブルに対応できます。
- インストーラーの検討: 配布プラットフォームに関わらず（クラウドストレージを除く）、特に技術に詳しくないユーザーを対象とする場合は、アプリケーションをシンプルなインストーラー（Inno Setup や NSIS などを使用）でラップすることを検討すると、ユーザーエクスペリエンスが向上します。

7. 結論

Python (tkinter/pyinstaller) で作成された複数の小規模 Windows 実行可能ファイルを配布するには、様々な選択肢が存在します。GitHub Releases は、開発者が既に利用しているプラットフォームであり、ファイルサイズに関する懸念は誤解に基づく可能性が高く、実際には非常に有力な無料オプションです [7]。itch.io は、butler による効率的なアップデート機能や信頼性向上の可能性を提供し、優れたユーザー体験を実現する選択肢となります [17]。Microsoft Store は、最大のリーチと信頼性を提供しますが、登録料とアプリごとの提出・審査プロセスという大きなオーバーヘッドを伴います [24]。汎用クラウドストレージやセルフホスティングは、それぞれシンプルさやコントロール性を提供しますが、ソフトウェア配布に特化した機能やサポートは限定的です。

最終的に「最良の」方法は、開発者の労力、コスト、求めるユーザーエクスペリエンス、そして多数のツールを管理する上でのスケーラビリティといった要因のバランスによって決まります。まずは既存の GitHub アカウントを活用して Releases 機能を試し、その上で itch.io をいくつかのツールで実験的に導入してワークフローを比較検討することから始めるのが現実的かもしれません。ソースコードの共有から実行可能バイナリの配布へと移行することは、アプリケー

ションのアクセシビリティを高める上で重要な一歩です。

引用文献

1. Python Program Distribution and Installation Methods | Server As Code Dot Com, 4月 19, 2025にアクセス、https://serverascode.com/2024/12/21/python-installation-methods.html
2. Using PyInstaller to Easily Distribute Python Applications - Real Python, 4月 19, 2025にアクセス、https://realpython.com/pyinstaller-python/
3. PyInstaller : Everything you need to know about this Python application - DataScientest.com, 4月 19, 2025にアクセス、https://datascientest.com/en/pyinstaller-everything-you-need-to-know-about-this-python-application
4. What PyInstaller Does and How It Does It, 4月 19, 2025にアクセス、https://pyinstaller.org/en/stable/operating-mode.html
5. Using PyInstaller, 4月 19, 2025にアクセス、https://pyinstaller.org/en/v4.1/usage.html
6. 【PyInstallerの使い方】Pythonでの実行ファイル作成（exe化）方法を徹底解説！, 4月 19, 2025にアクセス、https://pythonsoba.tech/pyinstaller/
7. docs.github.com, 4月 19, 2025にアクセス、https://docs.github.com/repositories/working-with-files/managing-large-files/about-large-files-on-github
8. About releases - GitHub Docs, 4月 19, 2025にアクセス、https://docs.github.com/en/repositories/releasing-projects-on-github/about-releases
9. About releases - GitHub Docs, 4月 19, 2025にアクセス、https://docs.github.com/repositories/releasing-projects-on-github/about-releases
10. About Git Large File Storage - GitHub Enterprise Cloud Docs, 4月 19, 2025にアクセス、https://docs.github.com/enterprise-cloud@latest/repositories/working-with-files/managing-large-files/about-git-large-file-storage
11. Git LFS, 4月 19, 2025にアクセス、https://git-lfs.com/
12. git-lfs/docs/spec.md at main - GitHub, 4月 19, 2025にアクセス、https://github.com/git-lfs/git-lfs/blob/main/docs/spec.md
13. git-lfs/git-lfs: Git extension for versioning large files - GitHub, 4月 19, 2025にアクセス、https://github.com/git-lfs/git-lfs
14. Configuring Git Large File Storage - GitHub Docs, 4月 19, 2025にアクセス、https://docs.github.com/en/repositories/working-with-files/managing-large-files/configuring-git-large-file-storage
15. Installing Git Large File Storage - GitHub Docs, 4月 19, 2025にアクセス、https://docs.github.com/en/repositories/working-with-files/managing-large-files/installing-git-large-file-storage
16. Windows builds · The itch.io app book, 4月 19, 2025にアクセス、https://itch.io/docs/itch/integrating/platforms/windows.html
17. How do I *actually* make my .exe file distributable? : r/gamedev - Reddit, 4月 19,

2025にアクセス、https://www.reddit.com/r/gamedev/comments/113t705/how_do_i_actually_make_my_exe_file_distributable/
18. Quickstart guide · The itch app manual, 4月 19, 2025にアクセス、http://docs.itch.ovh/itch/v23.6.1/integrating/quickstart.html
19. LuaBinaries download | SourceForge.net, 4月 19, 2025にアクセス、https://sourceforge.net/projects/luabinaries/
20. SourceForge: Compare B2B Software, Download, & Develop Open Source & Business Software, 4月 19, 2025にアクセス、https://sourceforge.net/
21. Lua Binaries - SourceForge, 4月 19, 2025にアクセス、https://luabinaries.sourceforge.net/
22. Distribution Maker - SourceForge, 4月 19, 2025にアクセス、https://sourceforge.net/projects/distribution-maker/files/Distribution%20Maker%202.1.0/Distribution%20Maker.exe/download
23. Can I submit app in Python to Win10 App Store? - Microsoft Q&A, 4月 19, 2025にアクセス、https://learn.microsoft.com/en-us/answers/questions/169550/can-i-submit-app-in-python-to-win10-app-store
24. How to Publish an App on Microsoft Store: A Step-by-Step Guide - Spyboy blog, 4月 19, 2025にアクセス、https://spyboy.blog/2024/12/22/how-to-publish-an-app-on-microsoft-store-a-step-by-step-guide/
25. Publish Windows apps and games to Microsoft Store, 4月 19, 2025にアクセス、https://learn.microsoft.com/en-us/windows/apps/publish/
26. Steps to open a Windows developer account in Partner Center - Learn Microsoft, 4月 19, 2025にアクセス、https://learn.microsoft.com/en-us/windows/apps/publish/partner-center/open-a-developer-account
27. Account types, locations, and fees - Windows apps | Microsoft Learn, 4月 19, 2025にアクセス、https://learn.microsoft.com/en-us/windows/apps/publish/partner-center/account-types-locations-and-fees
28. Create and manage submissions - UWP applications - Learn Microsoft, 4月 19, 2025にアクセス、https://learn.microsoft.com/en-us/windows/uwp/monetize/create-and-manage-submissions-using-windows-store-services
29. Python code to submit apps, add-ons, and flights - UWP applications | Microsoft Learn, 4月 19, 2025にアクセス、https://learn.microsoft.com/en-us/windows/uwp/monetize/python-code-examples-for-the-windows-store-submission-api
30. Is there additional fee for free app submit-ion on windows store market - Stack Overflow, 4月 19, 2025にアクセス、https://stackoverflow.com/questions/14438218/is-there-additional-fee-for-free-app-submit-ion-on-windows-store-market
31. Set app pricing and availability for MSIX app - Windows apps - Learn Microsoft, 4

月 19, 2025にアクセス、
https://learn.microsoft.com/en-us/windows/apps/publish/publish-your-app/msix/price-and-availability