paozip for Python

Python ソースコードを暗号化して、大切なコードを守るツール

ユーザーズマニュアル

バージョン 2.0.0

2026年2月

有限会社 パオ・アット・オフィス

https://www.pao.ac/

目次

1. はじめに

1.1 paozip for Python とは

paozip for Python は、Python のソースコードを暗号化するツールです。

暗号化すると .py ファイルが .pye ファイルになり、ソースコードは完全に読めなくなります。でも——ラッパー経由で普通に import して実行できます。

こんな時に便利：

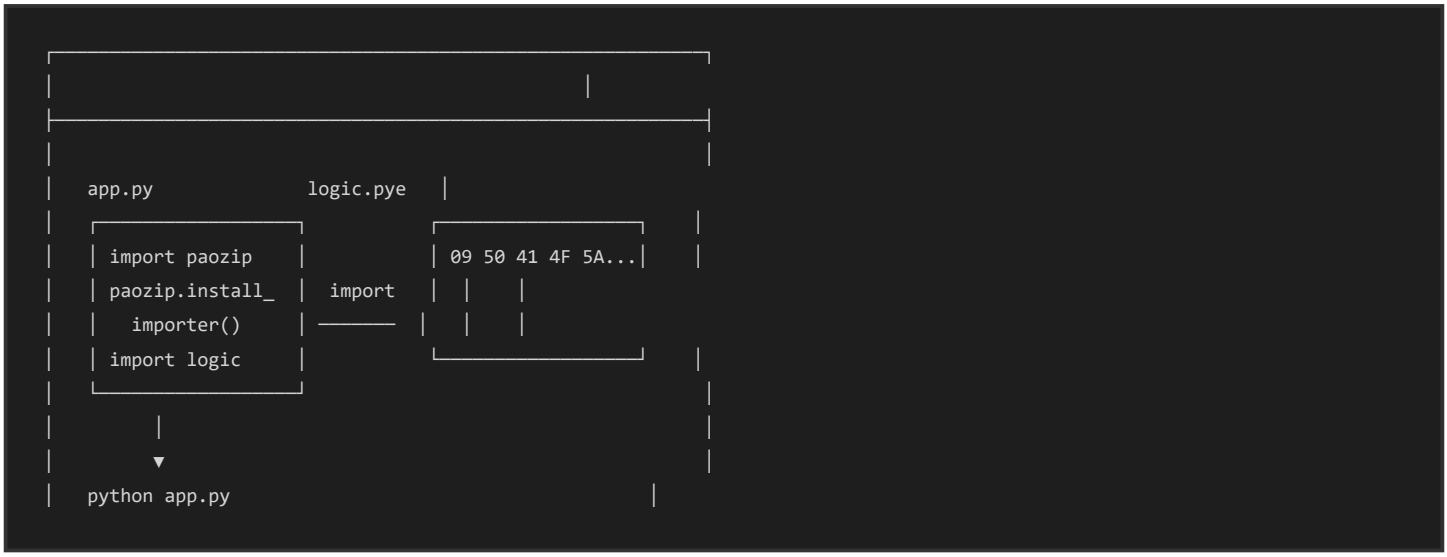
- 自社開発の Python アプリを納品したい（でもソースは見せたくない）
- AI/ML のモデル処理コードを配布したい（でもアルゴリズムは秘密にしたい）
- Django / Flask で作った業務システムを設置したい（でもロジックは隠したい）

1.2 特長

特長	説明
ラッパー方式	たった3行のラッパーを書くだけで暗号化ファイルが動く
import フック	import logic で .pye ファイルを自動的に復号して読み込み
CLI 直接実行	paozip run logic.pye でラッパーなしでも実行可能
C拡張で高速	暗号化エンジンは C言語実装。Python 純正並みの速度
圧縮効果	zlib 圧縮で、暗号化後のファイルサイズが小さくなることも
フレームワーク対応	Flask, Django, FastAPI などで使える

1.3 仕組み（ラッパー方式）

paozip for Python は ラッパー方式 で動作します。



```
│                                                    │
└────────────────────────────────────────────────────┘
```

ポイント：

- app.py（ラッパー）は公開してOK。中身はたった3行
- logic.pye（暗号化）に秘密のロジックを隠す
- ユーザーは python app.py で普通に実行できる
- ディスクには復号ファイルは書き出されない（メモリ上で復号）

1.4 PHP版との違い

```
PHP —                  Python —

sample.php           app.py→ logic.pye

php sample.php                      python app.py
     |                                   |
     ▼                                   ▼
                       import

: 1                    : 2
:                     : 3
: .php                  : .pye
```

比較	PHP版	Python版
暗号化方式	透過型（自動復号）	ラッパー型
必要なファイル数	1つ	2つ（ラッパー + 暗号化ファイル）
コード変更	不要	ラッパーに3行追加
拡張子	.php のまま	.pye に変更
直接実行	php file.php	paozip run file.pye

> Python は言語の仕組み上、PHP のような完全透過型は実現できません。
> でもラッパーはたった3行！ 十分シンプルです。

2. 動作環境

2.1 対応 OS

OS	対応
Ubuntu / Debian	
CentOS / RHEL / AlmaLinux	
Amazon Linux	
macOS	
Docker	

2.2 対応 Python バージョン

Python バージョン	対応
Python 3.8 〜 3.13+	

2.3 対応フレームワーク

フレームワーク	対応	備考
Flask		起動スクリプトの先頭で install_importer()
Django		manage.py の先頭で install_importer()
FastAPI		起動スクリプトの先頭で install_importer()
CLI アプリ		ラッパーまたは paozip run
その他		Python が動くフレームワークならすべて

3. セットアップ

3.1 Docker で試す（おすすめ）

Step 1: Docker 環境を起動

```
cd paozip-python-test
docker-compose up -d --build
```

初回はビルドに時間がかかります。コーヒーでも飲んで待ちましょう。

Step 2: コンテナに入る

```
docker-compose exec paozip-python-test bash
```

Step 3: セットアップ

```
cd /app/paozip-python
pip install .
```

Step 4: 確認

```
paozip version
```

バージョン情報が表示されれば成功！

3.2 Linux / macOS にインストール

必要なもの:

- Python 3.8 以上（開発ヘッダー含む）
- C コンパイラ（gcc）
- zlib

```
# Ubuntu/Debian
sudo apt install python3-dev gcc zlib1g-dev

# CentOS/RHEL
sudo dnf install python3-devel gcc zlib-devel
```

```
cd paozip-python
pip install .
```

3.3 インストールの確認

```
paozip version
python -c "import paozip; print(paozip.__version__)"
```

4. クイックスタート

4.1 暗号化してみよう

テスト用のファイルがあるとします：

```
# logic.py —
def secret_calculate(x, y):
    return (x * 137 + y * 42) ^ 0xDEAD

def get_message():
    return "Hello from encrypted Python!"
```

暗号化コマンド：

```
paozip encrypt logic.py
```

```
 logic.py → logic.pye
```

logic.py が logic.pye に暗号化されました。

```
cat logic.pye
# →
```

4.2 ラッパーを作ってみよう

暗号化したモジュールを使うには、ラッパーを書きます。たった3行です：

```
# app.py — OK
import paozip
paozip.install_importer()    # import

import logic                 # logic.pye

print(logic.get_message())
print(logic.secret_calculate(100, 200))
```

4.3 実行してみよう

```
python app.py
```

```
Hello from encrypted Python!
35065
```

暗号化されたコードが、ラッパー経由で普通に動きました！

> logic.py は暗号化されて logic.pye になっています。
> でも import logic と書くだけで、自動的に .pye ファイルから復号して読み込みます。

4.4 paozip run で直接実行

ラッパーを書かずに、暗号化ファイルを直接実行することもできます：

```
paozip run logic.pye
```

引数も渡せます：

```
paozip run my_script.pye --input data.csv --output result.json
```

> paozip run は、暗号化ファイルを復号してメモリ上で実行します。
> if __name__ == "__main__" のブロックも正しく動作します。

5. CLI コマンドリファレンス

5.1 encrypt（暗号化）

```
paozip encrypt <file.py> [-o <output>] [-b] [-f]
```

オプション	説明
-o <output>	出力先を指定（デフォルト: .pye 拡張子）
-b	元ファイルを .pao としてバックアップ
-f	確認なしで上書き

```
paozip encrypt logic.py               # → logic.pye
paozip encrypt logic.py -b            # → logic.pye + logic.pao
paozip encrypt logic.py -o secret.pye # → secret.pye
```

5.2 decrypt（復号）

```
paozip decrypt <file.pye> [-o <output>] [-f]
```

```
paozip decrypt logic.pye              # → logic.py
paozip decrypt logic.pye -o restored.py
```

5.3 run（実行）

```
paozip run <file.pye> [args...]
```

```
paozip run my_script.pye
paozip run my_script.pye arg1 arg2
paozip run server.pye --port 8080
```

5.4 check（確認）

```
paozip check <file>
```

暗号化されていれば終了コード 0、されていなければ 1。

```
paozip check logic.pye && echo ""
paozip check logic.py  || echo ""
```

5.5 version（バージョン）

```
paozip version
```

5.6 コマンド一覧

コマンド	説明
paozip encrypt <file>	暗号化
paozip decrypt <file>	復号
paozip run <file>	暗号化ファイルを直接実行
paozip check <file>	暗号化チェック
paozip version	バージョン表示

6. Python API リファレンス

```
import paozip
```

6.1 encrypt()

```
paozip.encrypt(data: bytes|str, key: str = None) -> bytes
```

データを暗号化します。文字列を渡すと自動的に UTF-8 エンコードされます。

```
encrypted = paozip.encrypt("")
encrypted = paozip.encrypt(b"\x00\x01\x02")
```

6.2 decrypt()

```
paozip.decrypt(data: bytes, key: str = None) -> bytes
```

暗号化されたデータを復号します。戻り値は bytes です。

```
decrypted = paozip.decrypt(encrypted)
text = decrypted.decode('utf-8')  #
```

6.3 encrypt_file()

```
paozip.encrypt_file(input_path: str, output_path: str = None, key: str = None) -> str
```

ファイルを暗号化します。出力パスを省略すると .pye 拡張子になります。

```
paozip.encrypt_file("logic.py")             # → logic.pye
paozip.encrypt_file("logic.py", "out.pye")  # → out.pye
```

6.4 decrypt_file()

```
paozip.decrypt_file(input_path: str, key: str = None) -> bytes
```

暗号化ファイルを復号して bytes を返します（ファイルには書き出しません）。

```
data = paozip.decrypt_file("logic.pye")
source = data.decode('utf-8')
```

6.5 decrypt_file_to()

```
paozip.decrypt_file_to(input_path: str, output_path: str = None, key: str = None) -> str
```

暗号化ファイルを復号してファイルに書き出します。

```
paozip.decrypt_file_to("logic.pye")              # → logic.py
paozip.decrypt_file_to("logic.pye", "restored.py") # → restored.py
```

6.6 is_encrypted() / is_encrypted_file()

```
paozip.is_encrypted(data: bytes) -> bool
paozip.is_encrypted_file(path: str) -> bool
```

データやファイルが暗号化されているか判定します。

```
if paozip.is_encrypted_file("logic.pye"):
    print("")
```

6.7 install_importer()

```
paozip.install_importer() -> None
```

Python の import フックをインストールします。これを呼んだ後、import 文で .pye ファイルを自動的に読み込めるようになります。

```
import paozip
paozip.install_importer()

import my_module  # my_module.pye
```

> 重要： 必ずモジュールのトップレベルで呼んでください。
> if __name__ == "__main__" の中で呼ぶと、それ以前の import には効きません。

7. ラッパーの書き方

7.1 基本パターン

```
# app.py — OK
import paozip
paozip.install_importer()

#  import
import logic           # → logic.pye
import utils           # → utils.pye
from models import User  # → models/__init__.pye

#
logic.main()
```

7.2 Web アプリ（Flask / Django）

Flask:

```
# app.py
import paozip
paozip.install_importer()

from flask import Flask
import secret_routes  # secret_routes.pye

app = Flask(__name__)
secret_routes.register(app)

if __name__ == '__main__':
    app.run()
```

Django:

```
# manage.py
import paozip
paozip.install_importer()

#  Django
import os
import sys

def main():
    os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'myproject.settings')
    # ...
```

7.3 CLI アプリ

```
# run.py
import paozip
paozip.install_importer()
import my_cli  # my_cli.pye

if __name__ == '__main__':
    my_cli.main()
```

7.4 やってはいけないこと

NG: if __name__ の中で install_importer() を呼ぶ

```
# NG!
import paozip

import logic  # ← →

if __name__ == '__main__':
    paozip.install_importer()  # ←
    logic.main()
```

OK: トップレベルで呼ぶ

```
# OK!
import paozip
paozip.install_importer()  # ← import

import logic  # ←  .pye
```

> install_importer() は import 文よりも前に 呼んでください。
> これさえ守れば、あとは普通の Python と同じです。

8. 暗号化キーについて

8.1 キーの仕組み

暗号化キーは paozip のビルド（pip install）時に自動生成されます。

キーの優先順位：

優先度	ソース	説明
1	PAOZIP_KEY 環境変数	環境変数で明示指定
2	.paozip_key ファイル	ビルドディレクトリ内のキーファイル
3	自動生成	ランダムに生成（初回ビルド時）

キーは 64 文字の16進数文字列（256 ビット相当）です。

8.2 複数環境で使う場合

開発環境と本番環境で同じ暗号化ファイルを使うには、同じキーでビルドする必要があります。

方法 1: .paozip_key ファイルをコピー

```
#
scp paozip-python/.paozip_key user@production:/tmp/paozip-python/

#
cd /tmp/paozip-python
pip install .
```

方法 2: 環境変数で指定

```
export PAOZIP_KEY="6416"
pip install .
```

8.3 注意事項

- .paozip_key ファイルを紛失すると暗号化ファイルを復号できなくなります
- .paozip_key は 安全な場所に保管してください
- 本番環境に .paozip_key を配布する必要はありません（ビルド済みの paozip がキーを内包）
- .pye ファイルを Web サーバーの公開ディレクトリに置かないでください

9. 各種環境での利用

9.1 Flask

```
# app.py
import paozip
paozip.install_importer()

from flask import Flask
import secret_logic  # → secret_logic.pye

app = Flask(__name__)

@app.route('/')
def index():
    return secret_logic.process()
```

```
paozip encrypt secret_logic.py
rm secret_logic.py     # .pye
python app.py
```

9.2 Django

```
# manage.py
import paozip
paozip.install_importer()
```

```
#
paozip encrypt myapp/views.py
paozip encrypt myapp/models.py
rm myapp/views.py myapp/models.py

# Django
python manage.py runserver
```

> .py と .pye の両方が存在する場合、.py が優先されます。
> 暗号化後は必ず元の .py ファイルを削除してください。

9.3 Docker

```
FROM python:3.12-slim

#
RUN apt-get update && apt-get install -y gcc zlib1g-dev

# paozip
COPY paozip-python/ /tmp/paozip-python/
RUN cd /tmp/paozip-python && pip install .

#
COPY app/ /app/

#
RUN paozip encrypt /app/secret_logic.py \
 && rm /app/secret_logic.py

CMD ["python", "/app/app.py"]
```

10. トラブルシューティング

症状	対処法
ModuleNotFoundError: No module named 'logic'	install_importer() が import の前に呼ばれているか確認。.pye ファイルが sys.path 上にあるか確認
.py と .pye の両方がある場合	.py が優先されます。暗号化後は .py を削除してください
RuntimeError: decryption failed	暗号化した環境と実行環境で同じキーを使っているか確認
install_importer() が効かない	if __name__ の中ではなく、トップレベルで呼んでいるか確認
ビルドエラー	python3-dev, gcc, zlib1g-dev がインストールされているか確認
paozip コマンドが見つからない	pip install . が成功しているか確認。pip show paozip で確認

11. 使用許諾

paozip for Python の使用について、paozip for Python の使用者（以下「利用者様」と称します）と有限会社パオ・アット・オフィス（以下「弊社」と称します）は、以下の各項目についての内容に同意するものとします。

1. 使用許諾書

この使用許諾書は、利用者様が paozip for Python を使用する場合に同意しなければならない契約書です。

2. 使用許諾書の同意

利用者様が paozip for Python を使用する時点で、本使用許諾書に同意されたものとします。同意されない場合は、paozip for Python を使用する事はできません。

3. ライセンス（使用権）の購入

利用者様が paozip for Python の製品版を使用して開発を行う場合には、1台の開発用コンピュータで paozip for Python を使用するにあたり、1ライセンスを購入する必要があります。

お客様環境等、開発コンピュータでないマシンで paozip for Python を使用する場合ライセンスは必要ありません。ランタイムライセンスフリーでございます。

4. 著作権

paozip for Python の著作権は、いかなる場合においても弊社に帰属いたします。

5. 免責

paozip for Python の使用によって、直接的、あるいは、間接的に生じた、いかなる損害に対しても、弊社は補償賠償の責任を負わないものとします。

6. 禁止事項

paozip for Python 及びその複製物を第三者に譲渡・貸与する事は出来ません。paozip for Python を開発ツールとして再販/再配布することを禁止します。なお、暗号化されたファイルを配布することは問題ございません。

7. 保証の範囲

弊社は paozip for Python の仕様を予告無しに変更することがあります。その場合の利用者様に対する情報提供は、弊社 WEB サイトにて行う事とします。

8. 適用期間

本使用許諾条件は利用者様が paozip for Python を使用した日より有効です。

12. 代金支払い方法

paozip for Python の製品版をご利用頂ける場合は、ライセンスを購入して頂く必要があります。

体験版について: すべての機能を制限なくお試しいただけます（機能制限なし）。

必要なライセンス数の数え方

paozip for Python で開発を行うパソコンの台数

1ライセンス当たりの価格

11,000円（税込）

バグフィックス等のバージョンアップは原則として無償とさせて頂きます。

お支払方法

11,000円 × ライセンス数 を下記口座へ銀行振込、または、郵便振替による送金をして下さい。

銀行名	支店名（コード）	口座番号	名義
三菱UFJ銀行	新宿支店 (341)	普通 3831891	ユ）パオアットオフィス
PayPay銀行	すずめ支店 (002)	普通 6461359	ユ）パオアットオフィス

郵便口座番号	名義
00150-0-576845	有限会社 パオ・アット・オフィス

※ 振込手数料は利用者様負担でお願い致します。

お支払いの通知と製品の送付

1. 振り込みが完了した時点で、必ず弊社 WEB サイトの入金連絡フォームから入金のご連絡をお願いいたします。
2. 弊社では上記連絡を受けて入金確認を行い、paozip for Python の製品版を利用者様へ電子メールにてお送りさせて頂きます。

見積書/納品書/請求書/領収証の発行

見積書/納品書/請求書/領収証の発行は可能でございます。製品サイトでの手続きにより発行いたします。

お問い合わせ

製品に関するお問い合わせは、下記までお願いいたします。

有限会社 パオ・アット・オフィス

・製品サイト: https://www.pao.ac/paozip/

・購入ページ: https://www.pao.ac/paozip/buy.html

・メール: info@pao.ac

paozip for Python — あなたの Python コードを、シンプルに、確実に、守ります。