

# How to Use In-Memory Streams

Hayao Suzuki

PyCon JP 2020

August 28, 2020

# Who am I ?

お前誰よ

Name Hayao Suzuki (鈴木 駿)

Twitter @CardinalXaro

Work Python Programmer

# Who am I ?

## Reviewer of Technical Books

- Effective Python 第 2 版 (O'Reilly Japan)
- 動かして学ぶ量子コンピュータプログラミング (O'Reilly Japan)

# Who am I ?

## Talks

- Symbolic Mathematics using SymPy(PyCon JP 2018)
- Elementary Number Theory with Python(PyCon mini Hiroshima 2019)
- Do you know cmath module?(PyCon mini Shizuoka)

Lists are here <https://xaro.hatenablog.jp/>

## In-Memory Streams

# Stream?

そもそもストリームって何？

ストリームはファイルオブジェクトである。

# File Object?

## ファイルオブジェクトって何？

- `read()` や `write()` などのメソッドを持つオブジェクト
- ディスク上のファイルや別の場所にあるストレージ、入出力機器とやりとりができる

# File Object?

## ファイルオブジェクトたち

- 生バイナリファイル
- バッファ付きバイナリファイル
- テキストファイル



# 使い方

## テキストファイル

```
f = open("myfile.txt", "r")
```

## バッファ付きバイナリ

```
f = open("myfile.jpg", "rb")
```

# open 関数の裏側

open は何をしているのか？

OS のシステムコール API を呼ぶ

# open 関数の裏側

## 例：CSV に加工する

```
with open("events.csv", "w") as csv_file:
    fieldnames = ["title", "started_at", "ended_at"]
    writer = csv.DictWriter(csv_file, fieldnames)
    writer.writeheader()
    writer.writerows(events)
```

## 例 : Windows

- CreateFile (ファイルのアクセス権取得)
- QueryAllInformationFile (ファイル情報の取得)
- WriteFile (ファイルへ書き込む)
- CloseFile (ファイルを閉じる)

Process Monitor 経由で確認した。

## 例 : Ubuntu on WSL

- openat (ファイルのオープン)
- fstat (ファイル情報の取得)
- ioctl (デバイス制御)
- lseek (ファイルのシーク)
- write (ファイルへ書き込む)
- close (ファイルを閉じる)

strace 経由で確認した。

# 最後に笑うのは誰だ

## 最終的な成果物はどこに置く？

- ファイルをローカルに保存するのがゴールではない
- ファイルを AWS S3 などの外部に置きたい

# In-Memory Streams

# インメモリーストリーム

## インメモリーストリームとは

- str や bytes をファイルオブジェクトのように扱える
- 読み書き可能、ランダムアクセス可能



# StringIO

## StringIO

テキストファイルのためのインメモリストリーム

# StringIO

## 例：CSV を StringIO で取り扱う

```
import io
with io.StringIO() as csv_file:
    fieldnames = ["title", "started_at", "ended_at"]
    writer = csv.DictWriter(csv_file, fieldnames)
    writer.writeheader()
    writer.writerows(events)
```

# BytesIO

## BytesIO

バッファ付きバイナリファイルのためのインメモリストリーム

## 例：PNG を BytesIO で取り扱う

```
import io
with io.BytesIO(png_bytes) as f:
    png_header = f.read(8)
    print(png_header)    # b'\x89PNG\r\n\x1a\n'
```

# ケーススタディ

## 取得、圧縮、アップロード

- データをインターネット経由で取得する
- データを加工する
- データを圧縮する
- 圧縮データを外部ストレージにアップロードする

# データをインターネット経由で取得する

例 : Connpass API をコールする

```
with urllib.request.urlopen(url) as response:  
    events = json.load(response)["events"]
```

# データを加工する

## 例：API の取得結果を CSV にする

```
with io.StringIO() as ts:  
    header = ["title", "started_at", "ended_at"]  
    writer = csv.DictWriter(ts, fieldnames=header)  
    writer.writeheader()  
    writer.writerows(events)
```

# データを圧縮&アップロード

例：ZIP に圧縮して AWS S3 にアップロード

```
with io.BytesIO() as bs:
    with zipfile.ZipFile(bytes_stream, "w") as zf:
        zf.writestr("events.csv", ts.getvalue())
    bs.seek(0) # ファイルシークがポイント
    s3.upload_fileobj(bs, "bucket", "events.zip")
```