SRA の技術情報誌
GS LetterNeo
Vol.176　2023 年 3 月

PyScript を使った Python による Web アプリケーション作成

熊澤 努

技術本部 先端技術研究室

はじめに

今回の記事では、Python で書かれたプログラムを Web ブラウザ上で実行するフレームワークである PyScript[1]を紹介します。PyScript を使うことで、Python プログラムを HTML ファイルに埋め込むことができるだけでなく、Microsoft Edge などの既存のブラウザで簡単に実行することができます[2]。

PyScript の準備

PyScript を使うためには、専用の Python パッケージをインストールする必要はありません。ただし、筆者は、Visual Studio Code を開発環境として使用し、その拡張機能である Live Server(Five Server)[3]を使って動作を確認しました。

[1] https://pyscript.net/

[2] この記事のプログラムの動作は 2022.12.1 版で確認しています。なお、公式サイトによると、この記事を執筆した 2023 年 3 月の時点で公開されている PyScript はアルファ版であり、製品やサービス開発用ではないとされています。

[3] https://marketplace.visualstudio.com/items?itemName=yandeu.five-server

HTML を作成する

簡単な例として、ランダムグラフをブラウザで表示するプログラムを作ります。ここでは、複雑ネットワーク分析で使われる Python パッケージである networkx[4]を使い、Erdős-Rényi グラフ[5]を描画します。Erdős-Rényi グラフは、頂点の個数とそれらの間の接続確率が与えられたときに、頂点同士を接続確率に従ってランダムに接続したグラフ構造です。Erdős-Rényi グラフそのものは、networkx の erdos_renyi_graph API[6] で簡単に作ることができます。なお、生成したグラフの描画には、Python の可視化用パッケージである matplotlib[7]を使います。

まず、HTML ファイルを作成します。以下に作成する HTML の構成を示します。random_graph.html という名前で保存しておきましょう。

```
<html>
  <head>
    <meta charset="utf-8">
    <link rel="stylesheet" href="https://pyscript.net/latest/pyscript.css"/>
    <script defer src="https://pyscript.net/latest/pyscript.js"></script>
  </head>
  <body>
    <py-config>
          <!-- 依存関係やインポートするパッケージなどのメタデータを記述 -->
    </py-config>
    <py-script>
        <!-- Pythonプログラムを記述 -->
    </py-script>
  </body>
</html>
```

pyscript.css と pyscript.js を読み込むことで、ブラウザ上で Python プログラムを実行することができます。py-config タグと py-script タグは PyScript が使用するタグです。py-config タグにはメタデータを記述します。ここで記述するメタデータには、Python プログラムがインポートするパッケージや、呼び出したいローカルファイルの情報などがあります。py-script タグには、Python でプログラムを記述します。

---

[4] https://networkx.org/

[5] Erdős-Rényi グラフの詳しい説明は省略します。例えば以下の Wikipedia の記事を参考にしてください。

https://en.wikipedia.org/wiki/Erd%C5%91s%E2%80%93R%C3%A9nyi_model

[6] https://networkx.org/documentation/stable/reference/generated/networkx.generators.random_graphs.erdos_renyi_graph.html

[7] https://matplotlib.org/

Python プログラムを追加する

random_graph.html に Python を動かすための記述を追加していきます。最初にメタデータです。今回は matplotlib と networkx を使うので、py-config タグに両パッケージを宣言します。

random_graph.html

```
<html>
  ...
  <body>
    <py-config>
        packages = [
            "matplotlib",
            "networkx"
        ]
    </py-config>
    ...
  </body>
</html>
```

次に、py-script タグに Python プログラムを記述します。

random_graph.html

```
<html>
  ...
  <body>
    ...
    <py-script>
        import matplotlib.pyplot as plt
        import networkx as nx
        from pyscript import Element

        def er_draw():
            nodes = int(Element('nodes').element.value)
            probability = float(Element('probability').element.value)
            print(f'nodes={nodes}, probability={probability}')

            fig, ax = plt.subplots()

            er_graph = nx.erdos_renyi_graph(nodes, probability)
            nx.draw(er_graph, ax=ax)
            display(fig, target='graph_area')
    </py-script>
  </body>
</html>
```

後でボタンを押したときのイベント処理にするため、グラフを作成して描画する処理は関数 er_draw にまとめています。また、頂点数と接続確率はブラウザ上でユーザが入力できるようにしたいので、テキストボックスから取得することにします。それぞれ"nodes"と"probability"という id のテキストボックスとします。テキストボックスの入力値を取得するには、PyScript の Element API を使用します。Element を使用すると、指定した id の値に該当する DOM の要素を操作することができます。上のプログラムでは、テキストボックスの値を文字列で取得して、数値に変換しています（簡単のためエラーチェック等は省略しています）。networkx の draw API で生成したグラフを描画した後、表示は PyScript の display API を

使って、ブラウザ上で行います。display API の引数 target には表示領域の id を指定します。表示領域は後で HTML ファイルに追加します。

HTML を完成させる

最後にブラウザに配置する GUI 部品を HTML で記述します。今回は、ユーザが頂点数と接続確率を入力するためのテキストボックス ("nodes、"probability")、グラフの生成と描画を実行するイベントを発行するボタン ("er-draw")、グラフの表示領域 ("graph-area") を HTML のタグで以下のように指定します。

random_graph.html

```
<html>
  ...
  <body>
    <div id="graph_area"></div>
    ...
    <div>
        <input id="nodes" class="py-input" value='20' type="text">
        <input id="probability" class="py-input" value='0.4' type="text">
        <button id="er-draw" class="py-button" type="submit" py-
click="er_draw()">
        Draw Erdos-Renyi Graph
        </button>
      </div>
  </body>
</html>
```

テキストボックスとボタンのクラス名とクリックイベントの属性名には、それぞれ"py-input"、py-click と PyScript 独自の値を指定する必要があります（スペースの都合のため上のプログラムでは py-click が途中で改行されています）。また、py-click には er_draw()と Python の関数呼び出しを設定します。この設定により、ユーザからのボタンクリックイベントで er_draw が実行されるようになります。以上で完成です。

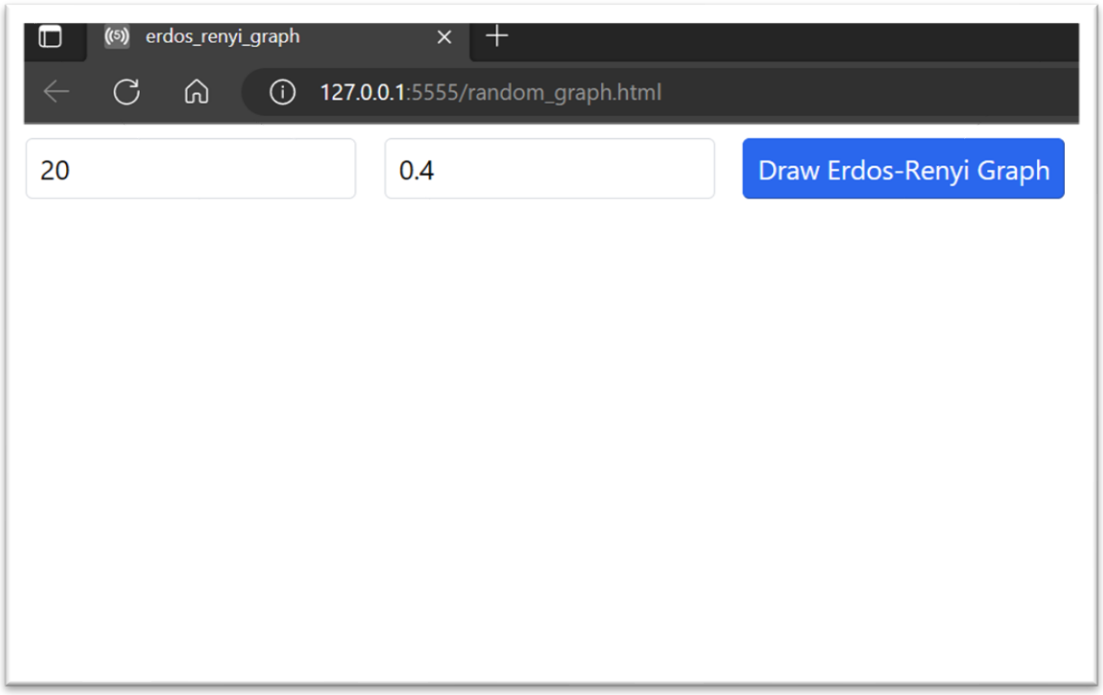
完成した HTML ファイル全体を以下に示します。

random_graph.html

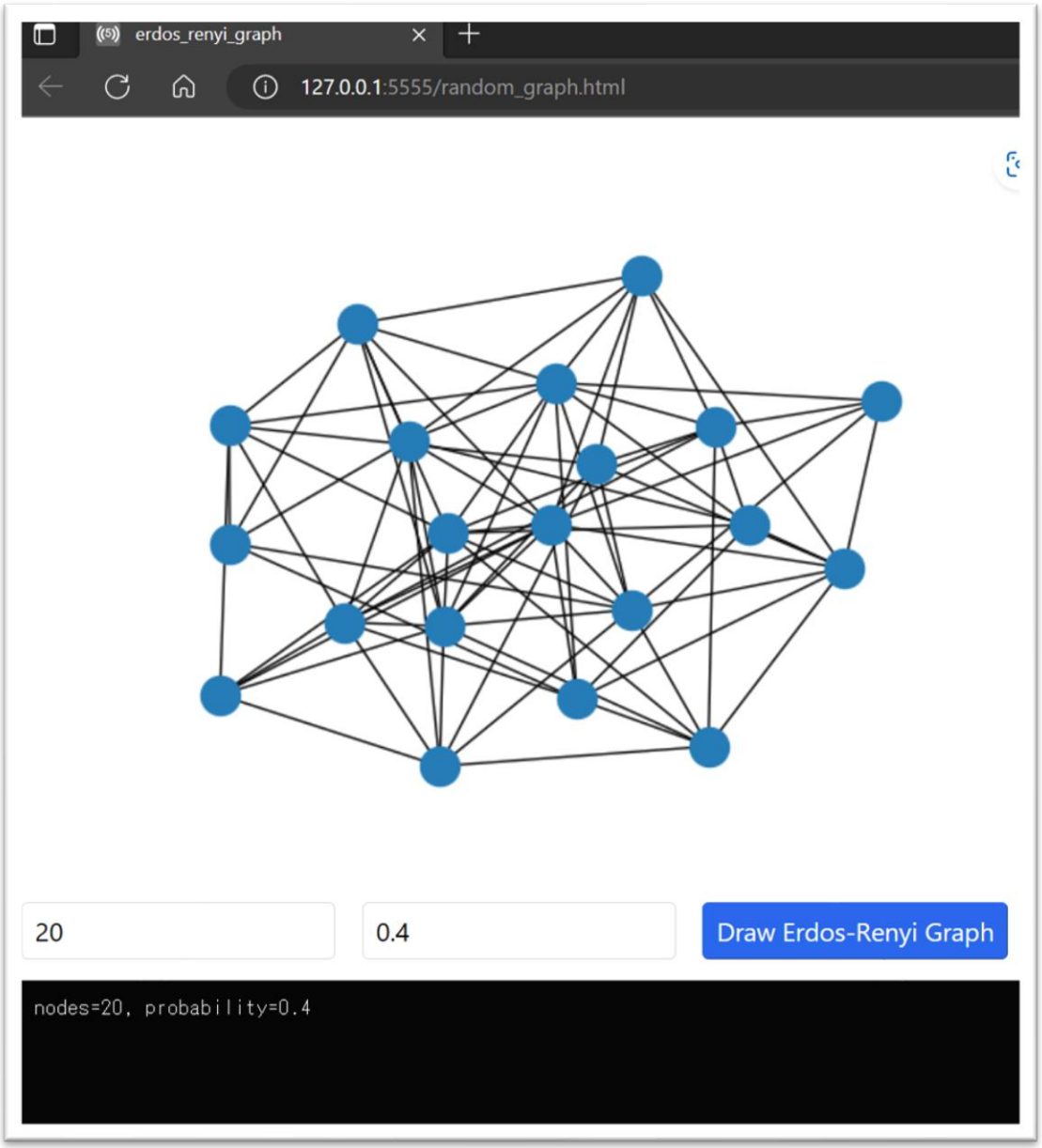
```
<html>
  <head>
    <title>erdos_renyi_graph</title>
    <meta charset="utf-8">
    <link rel="stylesheet" href="https://pyscript.net/latest/pyscript.css"/>
    <script defer src="https://pyscript.net/latest/pyscript.js"></script>
  </head>
  <body>
    <div id="graph_area"></div>
    <py-config>
        packages = [
            "matplotlib",
            "networkx"
        ]
    </py-config>
    <py-script>
        import matplotlib.pyplot as plt
        import networkx as nx
        from pyscript import Element

        def er_draw():
            nodes = int(Element('nodes').element.value)
            probability = float(Element('probability').element.value)
            print(f'nodes={nodes}, probability={probability}')

            fig, ax = plt.subplots()

            er_graph = nx.erdos_renyi_graph(nodes, probability)
            nx.draw(er_graph, ax=ax)
            display(fig, target='graph_area')
    </py-script>
    <div>
        <input id="nodes" class="py-input" value='20' type="text">
        <input id="probability" class="py-input" value='0.4' type="text">
        <button id="er-draw" class="py-button" type="submit" py-click="er_draw()">
        Draw Erdos-Renyi Graph
        </button>
      </div>
  </body>
</html>
```

スクリプトを実行する

完成した HTML を実行してみましょう。ブラウザで random_graph.html を開いて暫く待つと以下のように表示されます。



特に何も書かれていませんが、左のテキストボックスが頂点数、右のテキストボックスが接続確率です。とりあえずこのままにして、"Draw Erdos-Renyi Graph"ボタンを押すと、以下のようにグラフがブラウザ上に表示されました。なお、Python プログラム中の print 文による出力は、PyScript が生成した専用の領域に表示されます。



おわりに

今回は、Python プログラムを埋め込んだ HTML ファイルを Web ブラウザ上で実行するフレームワークである PyScript を紹介しました。今回は紹介できませんでしたが、PyScript には JavaScript と相互にオブジェクトをやり取りする機能があり、既存の Web アプリケーション開発とも親和性が高いと考えられます。また、開発者自身が作成した Python のローカルファイル（.py ファイル）を HTML から呼び出す機能もあり、従来の Python プログラマにも使いやすくなっています。ブラウザを使ったグラフィカルなプログラムには Jupyter Notebook や Google Colaboratory などで採用されているノートブック形式がよく知られていますが、PyScript も選択肢の一つに入れることができると思います。

GSLetterNeo Vol.176
2023 年 3 月 20 日発行
発行者　株式会社 SRA 技術本部 先端技術研究室

編集者　熊澤努　方 学芬
バックナンバー https://www.sra.co.jp/public/sra/gsletter/
お問い合わせ　gsneo@sra.co.jp

株式会社SRA
〒171-8513 東京都豊島区南池袋 2-32-8

夢を。Yawaraka Innovation
やわらかいのべーしょん