Python/flet を用いた GUI プログラミングの教育

箕原辰夫*1
Email: minohara@cuc.ac.jp

*1: 千葉商科大学政策情報学部（2025 年度よりサービス創造学部）

◎Key Words　　Python，flet，Flutter，GUI，GUI プログラミング

1.　はじめに

2025 年度春学期のゼミナールでのアプリケーション作成プログラミング環境として、Python/flet[1]を採用した理由は、Dart/Flutter[1]や、Swift/SwiftUI[3]を利用した宣言的な GUI（Graphic User Interface）プログラミングに対して、Python/flet による GUI プログラミングは、旧来の手続き型になっていて、プログラミングの授業においてPython を学んできた学生に教育するのに、教えやすいものとなっているからである。宣言的な GUI プログラミングでは、記号の特殊な使い方や、複数の括弧が組み合わさっていて、母体となるプログラミング言語に対して、追加の文法規則を随時教えていく必要がある。これに対して、Python 上の flet では、従来の手続き型になっているので、Python の文法を理解していれば、これまでのライブラリを利用する方式で、そのまま GUI を記述することが可能になっている。また、flet は Flutter と同様に、作成したアプリケーションを、スマートフォンや Web を含めて、複数のプラットフォーム用に作成することが可能であるため、学生が将来アプリケーション開発に進むときの経験を与えることができる。ここでは、2025 年春学期にゼミナールで行なった教育結果を元に flet の使い方および使用感について紹介する。

2.　flet の環境設定

2.1　インストールと実行

Windows や Mac OS、Linux を問わず、Python がインストールされていれば、以下のように pip を使ってインストールが可能である。

pip install flet (Windows では管理者ターミナルで)
sudo pip3 install flet（Mac OS や Linux で全体にインストールする場合)

実行は、以下の例のように Python のプログラムを開発環境のプラットフォームでローカルな Python インタープリタ上で実行する方式と Web で実行する方式の両方が可能になっている。

flet run program.py（ローカルに実行)
flet run --web program.py (Web ブラウザ上で実行)

また、Python 標準の Python IDLE や VSCode などの開発環境上では、通常の Python のプログラムとして実行すれば、自動的に、flet のインタープリタが起動され、ローカルに実行することが可能である。

2.2　アプリケーションの移行

他のクロスプラットフォームのライブラリと同様に、作成したアプリケーションを Android 携帯・タブレット用に移行（publish）したい場合は、Android/Java SDK が必要であり、iPhone/iPad 用に移行する場合は、Xcode と iOS SDK が必要となってくる。標準では、開発環境のプラットフォームに移行、および Android と Web 用にアプリケーションを移行することができる。それぞれのプラットフォーム用にアプリケーションを移行・作成（publish/build）するには、次のようなコマンドを使って、アプリケーションを作成する。ただし、該当フォルダに作成に必要なプロジェクト構造を予め設定しておかなければならない[4]。

```
flet build apk　# Android APK（リリース）形式
flet build aab　# Android AAB（Google Play）形式
flet build ipa　# iOS App Archive 形式
flet build macos　# Mac OS 形式
flet build windows　# Windows 形式
flet build linux　# Linux 形式
flet build web　# Web 形式
```

それぞれの開発プラットフォームでは、以下の図のように、アプリケーションを移行できる環境は限られているので、目的の環境用にアプリケーションを移行・作成するには、開発環境を選ぶ必要がある。なお、図中の WSL は、Windows Subsystem for Linux の略である。

Run on / `flet build`	`apk/aab`	`ipa`	`macos`	`linux`	`windows`	`web`
macOS	✅	✅	✅			✅
Windows	✅			✅ (WSL)	✅	✅
Linux	✅			✅		✅

図 1 移行・作成できる環境一覧[4]

3. GUI の記述方法

ゼミナールでは、13 回の授業回数に渡って、30 本ぐらいのプログラムを記述し、実行する予定にしている（この論文の提出時は、既に 20 本のプログラムを記述している）。以下に、一番簡単なアプリケーションの記述から始めて、実際に教育を行なう上で躓きそうになる部分を中心にプログラミングの記述方法を紹介していく。

3.1 ウィンドウの記述方法

アプリケーションのトップウィンドウは、通常はページ（Page クラスのオブジェクト）という形で記述し、メインの関数に渡される。ページの幅や高さなどは、関数側で渡されたページが持つウィンドウのプロパティとして設定する。flet では、オブジェクトの属性のことをプロパティと呼んでいる。プロパティの値を変更したら、オブジェクトに対して、update( )関数を呼び出す。

```
from flet import *
def main(page):
    page.window.width = 800
    page.window.height = 600
    page.update()
app(main)
```

3.2 コントロールの配置

flet では、GUI のコンポーネントのことをコントロール（Control）という名前で統一している。レイアウトと GUI コンポーネントの区別はない。レイアウトの場合は、最初の引数をリスト・タプルなどの形にして、レイアウトされるコントロールを指定する。なお、コントロールは最上位のものが、ページに追加される。ページに追加する場合は、add( )関数あるいは、ページの controls プロパティの append( )関数を用いる。以下のプログラムは、3 つのテキスト・コントロールを縦に並べて表示するものである。

```
from flet import *
def main(page):
    page.window.width = 800
    page.window.height = 400
    t1 = Text( value="Hello", color="green", size=70 )
    t2 = Text( value="日本語", color="red", size=64 )
    t3 = Text( value="ক্ষঘচ", color="blue", size=70 )
    columns = Column( [t1, t2, t3] )
    page.controls.append( columns )
    page.update()
app(main)
```

以降のプログラミング例では、import 文や main 関数定義、app()関数の呼出しの部分は共通なので省略し、page に追加するまでの部分だけを記述する。

3.3 レイアウト用のコントロール

レイアウト用のコントロールとしては、主に以下のようなものがある。flet には、レイアウトも含めて、iOS 用のコントロールが多く定義されているが、ここでは割愛する。

表 1 flet の主なレイアウト・コントロール

クラス	レイアウト内容
Column	縦方向のレイアウト
Row	横方向のレイアウト
DataTable	表形式のレイアウト
Card	カード形式のレイアウト
GridView	グリッド形式のレイアウト
ListView	リスト形式のレイアウト
Stack	スタック形式のレイアウト
Tabs	タブ形式のレイアウト

レイアウト上のコントロールの揃えも変えることができるが、主方向（MainAxisAlignment）と副方向（CrossAxisAlignment）で揃えを変えることができる。また、それぞれのレイアウトやコントロールごとに、揃えの指定のクラスが異なる（たとえば、Text コントロール内では、TextAlign を用いる）のは、少々面倒な感じを持った。レイアウトの細かな指定は、すべてそのレイアウト・コントロール用のクラスのプロパティとして設定することができる（プロパティとして用意してあれば）。

3.4 入力・選択用のコントロール

入力・選択用のコントロールとしては、通常の GUI ライブラリと同様で、ボタンを除くと主に以下のようなものがある。

表 2 flet の主な入力・選択用のコントロール

クラス	内容
Checkbox	チェックボックス
Dropdown	ドロップダウンメニュー
Radio	ラジオボタン
RangeSlider	範囲を指定するスライダー
SearchBar	検索のためのバー
Slider	スライダーによる数値入力
Switch	オン/オフのスイッチ
TextField	テキスト入力フィールド

それぞれのコントロールの入力・選択が行なわれると、イベントが発生し、そのコントロールの値（value プロパティのことが多い）を反映させるためのコールバック関数が呼ばれる。コールバック関数も、それぞれのコントロールのクラスのオブジェクトを作成する際に、キーワード引数で指定することができるし、プロパティなので、後から変更することも可能である。以下のプログラムの断片は、Slider での値の変更に応じて、コールバック関数を呼ぶための指定の例である。

```
slider = Slider( min=5, max=400, divisions=79, value=100,
```

label="{value}", on_change=changed)

コールバック関数では、event 引数を受け取り、その値を参照することが可能になっている。例えば、上記で指定されている change 関数は、以下のように定義することができる。event オブジェクトの control プロパティに、event を引き起こしたコントロール・オブジェクトが設定されているので、その value プロパティを参照する形で、値を参照することができる。また、上記の slider 変数が指しているスライダー・オブジェクトの value プロパティを参照することも可能である。

```
def changed( event ):
    print( event.control.value ) # slider.value でも良い
```

ボタンについては、Button クラスのコントロールを配置することによって可能であるが、形状や用途の違いにより、通常の Button クラス以外に、以下のようなボタンが用意されている。ここら辺は、昨今のスマートフォン用のアプリケーションを意識してのことだと思われる。

表 3 flet の主なボタン用のコントロール

クラス	内容
Button	ボタン
FilledButton	背景色付きボタン
FilledTonalButton	背景色付きアウトライン付きボタン
FloatingActionButton	浮遊するアクションボタン
IconButton	アイコンのみのボタン
MenuItemButton	メニューバー用のボタン
OutlinedButton	アウトライン付きのボタン
PopupMenuButton	ポップアップメニューを表示させるボタン
SegmentedButton	分割ボタン
SubmenuButton	サブメニューを表示させるボタン
TextButton	テキストのみのボタン

ボタンの場合は、作成の際に on_click キーワードでコールバック関数を呼び出す形になる。以下のプログラムの断片は、ボタンがクリックされたら、proceed という名前のコールバック関数を呼び出す記述になっている。

```
b = Button( text="OK", width=600, on_click=proceed )
```

3.5 情報表示用のコントロール

情報表示用のコントロールの中には、SwiftUI と同様に、Map クラスのオブジェクトがある。SwiftUI と異なるのは、SwiftUI では、Apple 独自のマップを用いるが、flet の場合は、OpenStreetMap[5]のタイルマップを用いている。

表 4 flet の主な情報表示用のコントロール

クラス	内容
Canvas	グラフィックス表示
CircleAvatar	ユーザのプロファイル画像表示
Icon	アイコン表示
Image	画像表示
Markdown	マークダウン(.md) 形式のテキスト表示
Map	地図表示
ProgressBar	進捗バー
ProgressRing	進捗リング
Text	テキスト表示
WebView	Web 上のページを表示
Lottie	Lottie[6]形式のアニメーション
Rive	Rive[7]形式のアニメーション

Canvas クラスのオブジェクトによるコントロールでは、通常のグラフィックス・ライブラリのように、それぞれのプリミティブ図形の描画用のオブジェクトが用意されている。以下のプログラムの断片は、描画色などを設定して、Pie 状の形状を Canvas 上に表示する記述になっている。なお、Canvas の一連のクラスは、flet.canvas モジュールに定義されている。

```
from flet.canvas import *
p = Paint( style=PaintingStyle.FILL, color=Colors.RED )
pie = Arc( x=75, y=50, width=250, height=250,
        start_angle=math.radians( angle ),
        sweep_angle=math.radians( 270 ),
        use_center=True, paint=p )
c = Canvas( [pie], width=400, height=300 )
page.add( c )
```

Bezier 曲線が使える Path クラスも用意されているが、決定的に問題があるのは、Canvas のグラフィックス・ライブラリでは、個々の Path に対して、移動・回転・拡大縮小・鏡像などのアフィン変換を行なえないことにある。

画像は、Web からでも、ローカルファイルからでも Image コントロールを使って読み込んで、表示させることができる。以下は、InteractiveViewer を使って、画像の拡大・縮小・移動などを、ピンチイン・アウトを用いてできるように記述したプログラムの断片になっている。

```
img = Image( src="https://picsum.photos/800/600",
        width=800, height=600 )
viewer = InteractiveViewer(content=img,
    boundary_margin=0, min_scale=1.0, max_scale=16.0 )
page.add( viewer )
```

地図表示の Map クラスのコントロールを使うには、flet_map のライブラリを pip などで追加インストールしなければならない。Map コントロールでは、ピンチイン・アウト、移動の他に、拡大縮小・地図の回転も可能になっている。表示では、OpenStreetMap のタイルレイヤー

（TileLayer）の上に、MarkerLayer、CircleLayer、PolygonLayer、そしてPolylineLayerなどのレイヤーを重ねて表示できるので、単なる地図表示だけではなくて、地図にあわせた情報表示も可能になっている。

```
import flet_map as mp
map = mp.Map( initial_center=
            mp.MapLatitudeLongitude(35.742, 139.909),
            initial_zoom=16,
            layers=[ mp.TileLayer(url_template="https://"+
                "tile.openstreetmap.org/{z}/{x}/{y}.png")],
            width=800, height=600 )
page.add( map )
```

WebViewや、LottieやRiveなどのアニメーション表示、グラフ・チャートの表示については、本稿の執筆時点では、まだ実習を行なっていないので、発表時にはその結果を含めて表示する。

3.6 マウス・キー入力

通常のGUIライブラリと同様に、マウスやタッチ入力は、GestureDetectorコントロールを用いて、パン開始・パン更新・パン終了用のコールバック関数を定義できる。また、ピンチイン・ピンチアウト・ピンチ移動については、前出のInteractiveViewerを用いる。キー入力については、スマートフォンでの仮想キーボードでのキー入力を意識してか、キー入力のコールバック関数しか用意されていない。また、pageのon_keyboard_eventプロパティに対して、コールバック関数を割り付ける形になる。

4. fletの使用感

4.1 学生と制作したアプリケーション

Canvasコントロールを用いたグラフィックを表示するアプリケーション以外に、翌月・前月・指定した年月を出すカレンダーを作成した。これは、DataTableレイアウトを用いて、各セル（Textコントロール）に、日付を入れる形になっている。flet標準では、DatePickerダイアログで同じような表示が可能となっているが、それよりかは、色合いを変えてみたりしたので、格好のアプリケーション作成サンプルとなった。

図2 完成したカレンダー・アプリケーション

4.2 他のGUIライブラリと比べて

Python標準のtkiner[9]ライブラリと較べると、スマートフォンのアプリケーション作成用に多彩なコントロールが用意されているが、スマートフォンを除くクロスプラットフォームで使えるPyQt5[10]ライブラリと較べると、特に、グラフィックス・ライブラリとしては、だいぶ見劣りのするものとなっている。特に、コントロールに対しては回転などもできる割には、アフィン変換がグラフィックスで使えないのは致命的とも言える。この点は、改良が必要と思われるので、GitHubにAffinePathなどのクラスを公開することも検討している。

5. おわりに

ゼミナールの学生は、Pythonのプログラミングの科目を既修あるいは、並行履修していたこともあり、DartやSwiftなどのプログラミング言語を新たに一から学ぶよりも、非常にスムーズに、fletの環境を利用してアプリケーションを作ることができた。タイピングの遅い学生もいたが、少人数のゼミナールということもあり、お互いにエラーが出たところを学生同士で教えあい、あるいは、教室のディスプレイに投影して、全員で確認したりして、FlutterやSwiftUIで行なった以上のGUIアプリケーションを作成することができた。ただ、スマートフォンに移行できるようになっていることから、PyQt5のような従来のPC主体のクロスプラットフォームのGUIライブラリに比べると、実現できないこともある。それに対して、スマートフォン対応のタッチなども標準でサポートしているところは、実際のスマートフォン用のアプリケーション開発には向いているだろう。今後は、引き続き、Python/fletを軸にゼミナールにおいてアプリケーション開発教育を展開していくことを予定している。

参考文献

(1) flet-dev, “Build multi-platform apps in Python powered by Flutter | Flet,” https://flet.dev
(2) Google, “Flutter - Build apps for any screen,” https://flutter.dev
(3) Apple Inc., “SwiftUI - Apple Developer,” https://developer.apple.com/jp/swiftui/
(4) flet-dev, “Publishing Flet app to multiple platforms,” https://flet.dev/docs/publish/
(5) OpenStreetMap Foundation, “OpenStreetMap,” https://www.openstreetmap.org
(6) Design Barn Inc., “LottieFiles: Download Free lightweight animations for website & apps,” https://lottiefiles.com/
(7) Rive Inc., “Rive — a new way to design, build, and ship user interfaces,” https://rive.app
(8) Plotly, “Plotly Open Source Graphing Library for Python,” https://plotly.com/python/
(9) Python, “tkinter — Python interface to Tcl/Tk,” https://docs.python.org/3.13/library/tkinter.html
(10) Riverbank Computing Limited, “PyQt5 Reference Guide — PyQt Documentation,” https://www.riverbankcomputing.com/static/Docs/PyQt5