Adatbányászat a Gyakorlatban

2. Gyakorlat: Dash alapok

Kuknyó Dániel
Budapesti Gazdasági Egyetem

2024/25
1.félév

Órai környezet telepítése

1. Új Anaconda környezet létrehozása `dash` néven:

```
$ conda create --name dash python=3.12
```

2. Környezet aktiválása:

```
$ conda activate dash
```

3. Órai tárhely klónozása:

```
$ git clone https://github.com/basictask/Adatbanyaszat.git
```

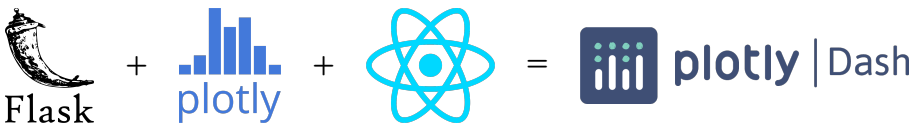
4. Függőségek telepítése:

```
$ cd Adatbanyaszat
$ pip install -r requirements.txt
```

A Dash keretrendszer

A Dash keretrendszer segítségével lehetséges interaktív, dinamikus adatalapú műszerfalakat és alkalmazásokat készíteni színtisztán Python nyelvben.

A Dash a **Flask** mikrokeretrendszert használja backend szerverként, **Plotly** segítségével jelenítui meg a diagramokat és **React** komponenseket használ a felhasználói interakció kezelésére.



A Dash könyvtár komponensei

Dash

Ez a fő csomag, amely bármely alkalmazás gerincét biztosítja a `dash.Dash` objektumon keresztül.

Emellett néhány más eszközt is biztosít az interaktivitás és kivételek kezeléséhez, amelyekről később fogunk beszélni, amikor építjük az alkalmazásunkat.

A Dash könyvtár komponensei

Dash Core Components

Egy csomag, amely interaktív komponensek készletét biztosítja, amelyeket a felhasználók manipulálhatnak.

Legördülő menük, dátumválasztók, csúszkák és sok más komponens is megtalálható ebben a csomagban.

A Dash könyvtár komponensei

Dash HTML Components

Ez a csomag az összes elérhető HTML címkét Python osztályként biztosítja. Egyszerűen átalakítja a Pythont HTML-re.

Például, Pythonban a `dash_html_components.H1('Hello, World')` kód átalakul `<h1>Hello, World</h1>` HTML kóddá.

A Dash könyvtár komponensei

Dash Bootstrap Components

Ez egy harmadik féltől származó csomag, amely Bootstrap funkcionalitást ad a Dash-hez. Ez a csomag és annak komponensei számos elrendezéssel és vizuális jelekkel kapcsolatos lehetőséget kezelnek.

Az elemek egymás mellé vagy egymás fölé helyezése, méretük meghatározása a böngésző képernyőmérete alapján, valamint kódolt színek készletének biztosítása a jobb kommunikáció érdekében a felhasználókkal.

Egy Dash alkalmazás struktúrája

- Importálások:

```
import dash
import dash_html_components as
    html
import dash_core_components as dcc
```

- Elrendezés:

```
app.layout = html.Div([
  dcc.Dropdown()
  dcc.Graph()
  ...
])
```

- Visszahívási függvények:

```
@app.callback()
...
@app.callback()
...
```

- Alkalmazás példányosítása:

```
app = dash.Dash(__name__)
```

- Alkalmazás futtatása:

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

Egy kezdeti alkalmazás

1. Egy új, app.py fájlban a következő csomagok importálásával:

```
import dash
import dash_core_components as dcc
```

2. Alkalmazás példányosítása:

```
app = dash.Dash(__name__)
```

3. Alkalmazás elrendezésének létrehozása:

```
app.layout = html.Div([
  html.H1('Hello, World!')
])
```

4. Futtatás:

```
if __name__ == '__main__':
  app.run_server(debug=True)
```

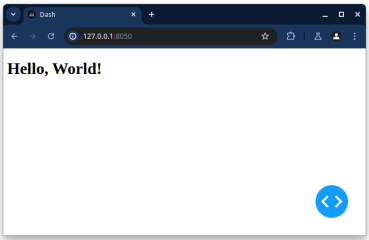
Az alkalmazás futtatása (app_v1_1.py)

A Python értelmező segítségével a megfelelő könyvtárban állva az app_v1_1.py fájl futtatásával az eredmény a következő:

```
(dash) daniel@neptune:~/Documents/BGE/
    Adatbanyaszat/2_dash/code$ python
    app_v1_1.py
Dash is running on http
    ://127.0.0.1:8050/

* Serving Flask app 'app_v1_1'
* Debug mode: on
```

A böngészőben a 127.0.0.1:8050 címre navigálva a következő eredmény látható:



Komponensek hozzáadása az alkalmazáshoz

Komponensek hozzáadása az alkalmazás elrendezésének szerkesztésével érhető el (`app.layout`). Ez meglehetősen egyszerű, csak a legfelső szintű `html.Div` komponens `children` attribútumához kell hozzáfűzni a megfelelő elemeket.

```
1 html.Div(children=[component_1, component_2, component_3, ...])
```

Div

A `Div` a HTML-ben egy blokk szintű elem, amely képes egy dokumentum különböző komponenseit csoportosítani. A `Div` nem rendelkezik semmilyen alapértelmezett stílussal vagy viselkedéssel.

HTML komponensek hozzáadása

`children`

Ez az első, és a fő konténere a komponenseknek. Paraméterül kaphatja elemek listáját vagy egyetlen elemet is.

`className`

Ez megegyezik a HTML `class` attribútumával.

`id`

A komponens azonosítója. Az interaktivitás megvalósításában van kulcsfontosságú szerepe

`style`

Ez megfelel az azonos nevű HTML attribútumnak azzal a különbséggel, hogy camelCase stílust használ a változók elnevezésére.

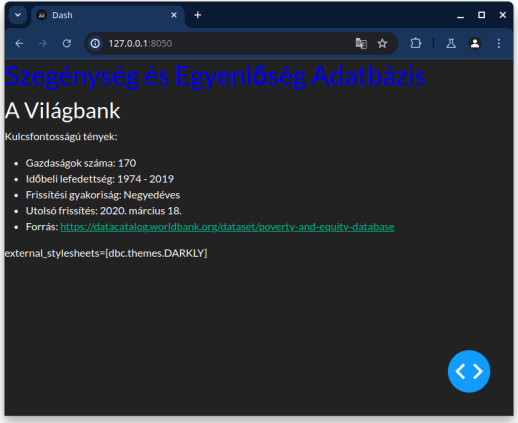
A Dash alkalmazás HTML komponensekkel (app_v1_2.py)

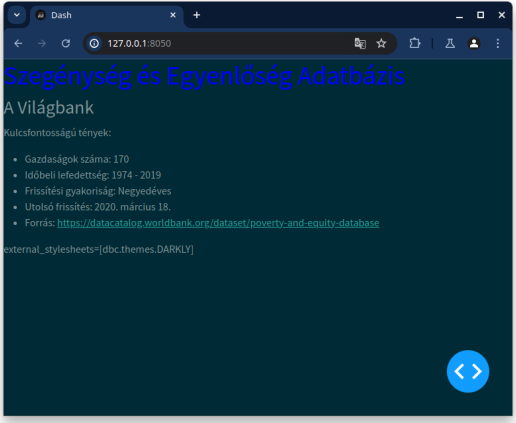
Témák

Egy Dash alkalmazás témájának megváltoztatása rendkívül egyszerű: a Dash objektum létrehozásakor kell egy új téma argumentumot bevinni a konstruktor függvénybe.

```
import dash_bootstrap_components as dbc
...
app = dash.Dash(__name__, external_stylesheets=[dbc.themes.BOOTSTRAP])
...
```

Témák az előző alkalmazásban (app_v1_3.py)

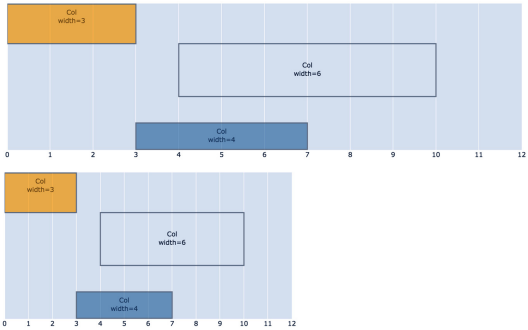




A rács rendszer

A Bootstrap segítségével lehetséges oszlopokat definiálni, ami egy független képernyőként viselkedik, egymás fölött megjelenítve az elemeket.

A rács rendszer 12 oszlopra bontja a képernyőt, és egy komponens szélessége oszlopok számában adható meg.

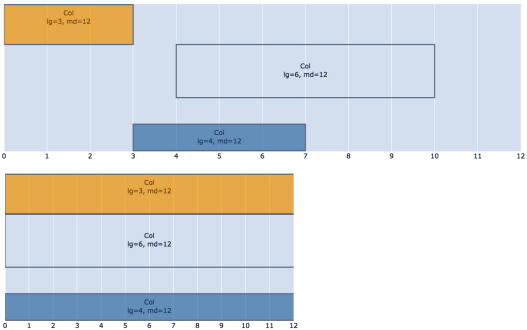



```
1 import dash_boostrap_components as dbc
2 dbc.Col(children=[child1, child2, ...])
```

Rácsok dinamikus képernyő méreten

Vannak olyan esetek, amikor nem kívánatos az elemek méretezése a képernyővel együtt. Amikor a képernyő kisebb lesz, némelyik komponenseknek jó, ha kiterjednek méretükben.

Öt különböző méretet lehet definiálni: `xs` (extra-small), `md` (medium), `lg` (large), `xl` (extra-large).

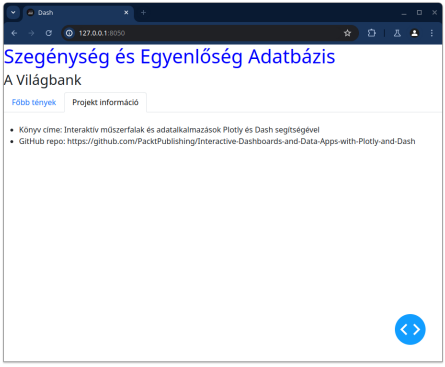



```
import dash_boostrap_components as dbc
dbc.Col(children=[child1, child2, ...], lg=6, md=12)
```

Bootstrap komponensek hozzáadása az alkalmazáshoz (app_v1_4.py)

Az alkalmazás következő verziójában két új komponens kerül hozzáadásra, a `Tabs` és `Tab`. Ezek szorosan kapcsolódnak egymáshoz. A `Tabs` a `Tab` konténere.

Ennek eredménye egy informatívabb és jobban elrendezett alkalmazás.



Dash alkalmazások Jupyter Notebookban

Kevés programkód változtatással az alkalmazás Jupyter Notebook környezetben is futtathatóvá válik. Ezt a `jupyter_dash` csomag teszi lehetővé.

A használatához a Dash helyett a JupyterDash csomagot kell importálni, és ennek mentén kell példányosítani az alkalmazást:

```
1 from jupyter_dash import JupyterDash
2 app = JupyterDash(__name__)
```

A JupyterDash három módot biztosít az alkalmazás futtatására:

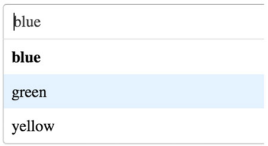
- `external`: Külön böngésző ablakban
- `inline`: A kód output helyen a notebookban
- `jupyterlab`: Külön böngészőfülben (csak JupyterLab szerveren)

Komponensek `id` paramétere

Az `id` paraméter elengedhetetlen a Dash alkalmazások interaktivitásához.

Ez egy, a komponensekhez rendelt egyedi azonosító, amelynek segítségével az alkalmazás megkülönbözteti és kezeli a különböző vezérlőelemeket, például grafikonokat vagy szövegdobozokat.

`id='color_dropdown'`



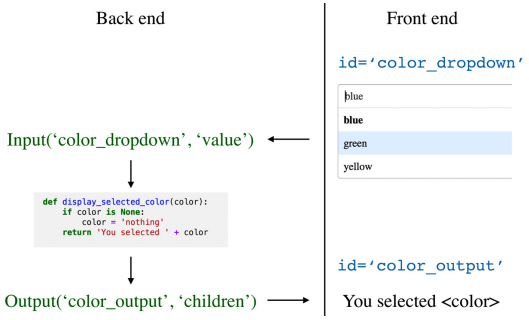

```
app.layout = html.Div([
  dcc.Dropdown(
    id='color_dropdown',
    options=[{'label': x, 'value': x} for x in ['blue', 'green', 'yellow']]
  )
])
```

Callback függvények

A callback egy Dash alkalmazásban egy olyan függvény, amely akkor hívódik meg, amikor egy adott esemény bekövetkezik, például egy felhasználói interakció.

Így dinamikusan frissíthetők az alkalmazás komponensei. A következők szükségesek egy callback függvényhez:

- `Output`: Az a komponens attribútum, amelyik meg fog változni a függvény hatására.
- `Input`: Az az alkalmazás elem vagy esemény, amelyik elindítja a függvényt.
- A függvény fejléc és definíció.



Callback függvény implementálása

A callback működés egy dekorátor segítségével valósítható meg a függvény fejléce fölött. Itt definiálni kell az `Input` és `Output` komponenseket, ilyen sorrendben.

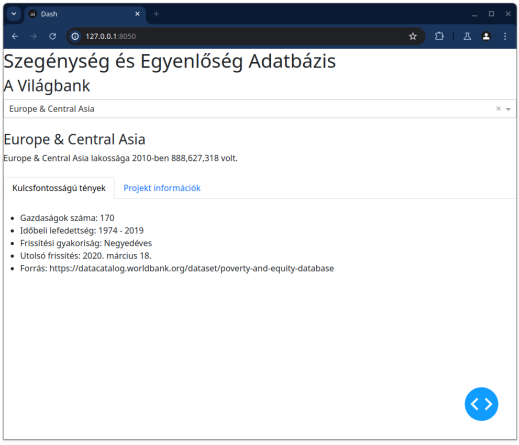
Egy callback függvénynek több inputja és outputja lehet.

```
@app.callback(
  Output('color_output', 'children'),
  Input('color_dropdown', 'value')
)
def display_selected_color(color):
  if color is None:
    color = 'nothing'
  return 'You selected ' + color
```

Callback függvény implementálása az alkalmazásba (app_v1_5.py)

Egy callback implementálásának lépései:

1. Új lenyíló lista létrehozása egy adathalmazban megtalálható országok segítségével
2. Egy új callback függvény létrehozása amely megkapja a választott országot, leszűri az adathalmazt majd megtalálja az ország népességi adatait (az összes forrásfájl a `data` mappában található).
3. Egy riport készítése a megtalált adatokról.



Callback függvények tulajdonságai

- A visszatérés előtt szinte bármilyen műveletet elvégezhetnek, mint pl. egy gépi tanulás modell tanítása
- A callback függvények harmadik attribútuma a (`State`). Az állapottal definiált objektum attribútumok nem indítják el a callback függvényt, de a futás során a függvény hozzáfér az értékükhöz.
- A callback dekorátor definíciós sorrendje: `[Output, Input, State]`.
- Az input és állapot sorrend a callback dekorátorban meg kell feleljen a paraméterek sorrendjének..