SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

Global Exploration Opportunities
GEO 2026

Data Visualization

Streamlit

Henry Novianus Palit

hnpalit@petra.ac.id

Agenda

- Streamlit installation and execution
- Page elements of streamlit (text, data, charts, widgets, layouts, status)
- Application logic of streamlit (navigation, pages, execution flow, caching, connection)

Installing **`streamlit`** *(1)*

Prerequisites:

- Python (version 3.9 to 3.13)
  - ☞ 3.12 will be used in this course
- Python package manager (e.g., **`pip`** or **`conda`**)
- Python environment manager (e.g., **`venv`** or **`conda env`**)
- Code editor / integrated development environment
  - ☞ VS Code will be used in this course

Installing **`streamlit`** *(2)*

Steps:

- Set up Python development environment
- Install streamlit
  - **`pip install streamlit`**
  - **`conda install streamlit`**
- Validate the installation by running
  - **`streamlit hello`**

Running **streamlit**

- Once your Python script program is ready, run it with
  - **`streamlit run your_script.py [-- script args]`**
- You can also pass a URL
  - **`streamlit run https://raw.githubusercontent.com/streamlit/demo-uber-nyc-pickups/master/streamlit_app.py`**

Page Elements [app01]: Write and Magic

```
import streamlit as st
import numpy as np
import pandas as pd

st.write("_Welcome_ to **FACT**!")

"_Welcome_ to **FACT**! :sunglasses:"

x = 10
'x', x  # 👈 Draw the string 'x' and then the value of x

df = pd.DataFrame(
    np.random.randn(10, 5),
    columns=('col %d' % i for i in range(5)),
)
df      # 👈 Draw the dataframe
```

Page Elements [app02]: Text Elements *(1)*

```
import streamlit as st

st.title(“:streamlit: is :blue[cool] :sunglasses:",
    anchor = False, help = "This is a title!")
st.header("This is a header with a divider",
    anchor = "Header", divider = "orange")
st.subheader("This is a subheader without a divider",
    anchor = "Subheader")
st.markdown('''
    :red[Streamlit] :orange[can] :green[write] :blue[text] :violet[in]
    :gray[pretty] :rainbow[colors] and :blue-background[highlight] text.
''')
st.markdown("Here's a bouquet &mdash;\
    :tulip::cherry_blossom::rose::hibiscus::sunflower::blossom:")
```

Emoji: https://share.streamlit.io/streamlit/emoji-shortcodes

Page Elements [app03]: Text Elements *(2)*

```
import streamlit as st

st.caption("A caption with _italics_ :blue[colors] and emojis :smiley:")

code = '''def hello():
    print("Hello, Streamlit!")'''
st.code(code, language = "python", line_numbers = True)

st.divider()    # 👈 Draws a horizontal rule

st.latex(r'''
    a + ar + a r^2 + a r^3 + \cdots + a r^{n-1} =
    \sum_{k=0}^{n-1} ar^k =
    a \left(\frac{1-r^{n}}{1-r}\right)
''')

st.text("This is text\n[**and** _more_ text].")
```

Page Elements [app04]: Dataframe *(1)*

```
import ...

df1 = pd.DataFrame(
    np.random.randn(10, 5),
    columns = ("col %d" % i for i in range(5)),
)
st.dataframe(df1.style.highlight_max(axis=0))

df2 = pd.DataFrame({
    "name": ["Roadmap", "Extras", "Issues"],
    "url": ["https://roadmap.streamlit.app",
            "https://extras.streamlit.app",
            "https://issues.streamlit.app"],
    "stars": [random.randint(0, 1000) for _ in range(3)],
    "views_history":
        [[random.randint(0, 5000) for _ in range(30)] for _ in range(3)],
})
```

(continued to next slide)

Page Elements [app04]: Dataframe *(2)*

```
st.dataframe(
    df2,
    column_config = {
        "name": "App name",
        "stars": st.column_config.NumberColumn(
            "Github Stars",
            help = "Number of stars on GitHub",
            format = "%d ★",
        ),
        "url": st.column_config.LinkColumn(
            "App URL"
        ),
        "views_history": st.column_config.LineChartColumn(
            "Views (past 30 days)", y_min = 0, y_max = 5000
        ),
    },
    hide_index = True,
)
```

Page Elements: Column Configuration

- `Column`
- `TextColumn`
- `NumberColumn`
- `CheckboxColumn`
- `SelectboxColumn`
- `DatetimeColumn`
- `DateColumn`
- `TimeColumn`
- `ListColumn`
- `LinkColumn`
- `ImageColumn`
- `AreaChartColumn`
- `LineChartColumn`
- `BarChartColumn`
- `ProgressColumn`

Column config: https://docs.streamlit.io/develop/api-reference/data/st.column_config

Page Elements: Column Configuration

Avatar	Name	Age	Is Active?	Activity (daily)	Homepage
	Gary Cross	82 years	☐		http://www.rivera.com/
	Madison Obrien	107 years	☐		https://nguyen.info/
	Ronald Miller	18 years	☐		https://king.com/
	Melanie Grant	96 years	☑		https://www.alvarez.com/
	Jaime Villegas	20 years	☑		https://www.jimenez.com/
	Amanda Smith	13 years	☑		https://reid-baxter.net/
	John Ellis	101 years	☑		https://mcmahon.biz/
	Amanda Griffin	7 years	☑		http://torres.org/
	David White	11 years	☑		https://www.hurst-bautista.
	Kelly Perkins	91 years	☐		http://greer.org/

Column config: https://docs.streamlit.io/develop/api-reference/data/st.column_config

Page Elements [app05]: Dataframe Selection

```
import streamlit as st
import pandas as pd
import numpy as np

if "df" not in st.session_state:
    st.session_state.df = pd.DataFrame(
        np.random.randn(10, 5),
        columns = ["a", "b", "c", "d", "e"],
    )

event = st.dataframe(
    st.session_state.df,
    key = "data",
    on_select = "rerun",
    selection_mode = ["multi-row", "multi-column"],
)

event.selection
```

Page Elements [app06]: Data Editor *(1)*

```
import streamlit as st
import pandas as pd

df = pd.DataFrame([
    {"command": "st.selectbox", "rating": 4, "is_widget": True},
    {"command": "st.balloons", "rating": 5, "is_widget": False},
    {"command": "st.time_input", "rating": 3, "is_widget": True},
])

edited_df = st.data_editor(
    df,
    column_config = {
        "command": "Streamlit Command",
```

(continued to next slide)

Page Elements [app06]: Data Editor *(2)*

```
        "rating": st.column_config.NumberColumn(
            "Your rating",
            help = "How much do you like this command (1-5)?",
            min_value = 1,
            max_value = 5,
            step = 1,
            format = "%d ⭐",
        ),
        "is_widget": "Widget ?",
    },
    num_rows = "dynamic",
    hide_index = True,
)

favorite_command = edited_df.loc[edited_df["rating"].idxmax()]["command"]
st.markdown(f"Your favorite command is **{favorite_command}** 👍")
```

Page Elements [app07]: Metric

```
import streamlit as st

a, b = st.columns(2)
c, d = st.columns(2)

a.metric(label = "Temperature",
         value = "30°F",
         delta = "-9°F",
         border = True)
b.metric("Wind", "4 mph", "2 mph", border = True)

c.metric("Humidity", "77%", "5%",
         delta_color = "off", border = True)
d.metric("Pressure", "30.34 inHg", "-2 inHg",
         delta_color = "inverse", border = True)
```

Page Elements [app08]: Area Chart

```
import streamlit as st
from vega_datasets import data

source = data.unemployment_across_industries()
source

st.area_chart(
    source,
    x = "date",
    y = "count",
    color = "series",
    stack = True,   # try also "normalize" and "center"
)
```

Page Elements [app09]: Bar Chart

```
import streamlit as st
from vega_datasets import data

source = data.barley()
source

st.bar_chart(
    source,
    x = "variety", y = "yield", color = "site",
    horizontal = True,
)

st.bar_chart(
    source,
    x = "year", y = "yield", color = "site",
    stack = False,
)
```

Page Elements [app10]: Line Chart

```
import streamlit as st
import pandas as pd
import numpy as np

chart_data = pd.DataFrame(
    np.random.randn(20, 3),
    columns = ["col1", "col2", "col3"],
)

st.line_chart(
    chart_data,
    x = "col1",
    y = ["col2", "col3"],
    color = ["#FF0000", "#0000FF"],
)
```

Page Elements [app11]: Scatter Chart

```
import streamlit as st
import pandas as pd
import numpy as np

chart_data = pd.DataFrame(
    np.random.randn(20, 3),
    columns = ["col1", "col2", "col3"],
)
chart_data["col4"] = np.random.choice(["A", "B", "C"], 20)

st.scatter_chart(
    chart_data,
    x = "col1",
    y = "col2",
    size = "col3",
    color = "col4",
)
```

Page Elements [app12]: Map

```
import streamlit as st
import pandas as pd
import numpy as np

df = pd.DataFrame({
    "col1": np.random.randn(200) / 50 + 37.76,
    "col2": np.random.randn(200) / 50 + -122.4,
    "col3": np.random.randn(200) * 100,
    "col4": np.random.rand(200, 4).tolist(),
})

st.map(
    df,
    latitude = "col1",
    longitude = "col2",
    size = "col3",
    color = "col4“,
)
```

Page Elements [app13]: Altair Chart *(1)*

```
import streamlit as st
import pandas as pd
import numpy as np
import altair as alt

if "data" not in st.session_state:
    st.session_state.data = pd.DataFrame(
        np.random.randn(20, 3), columns = ["a", "b", "c"],
    )
df = st.session_state.data

point_selector = alt.selection_point("point_selection")
interval_selector = alt.selection_interval("interval_selection")

chart = (
    alt.Chart(df)
        .mark_circle()
```

(continued to next slide)

Page Elements [app13]: Altair Chart *(2)*

```
        .encode(
            x = "a",
            y = "b",
            size = "c",
            color = "c",
            tooltip = ["a", "b", "c"],
            fillOpacity = alt.condition(
                point_selector, alt.value(1), alt.value(0.3)),
        )
        .add_params(point_selector, interval_selector)
)

event = st.altair_chart(chart, key = "alt_chart", on_select = "rerun")
event
```

Vega-Altair: https://altair-viz.github.io/gallery/

Page Elements [app14]: Vega-Lite Chart *(1)*

```
import streamlit as st
from vega_datasets import data

source = data.cars()

chart = {
    "mark": "point",
    "params": [
        {"name": "interval_selection", "select": "interval"},
        {"name": "point_selection", "select": "point"},
    ],
    "encoding": {
        "x": {
            "field": "Horsepower",
            "type": "quantitative",
        },
```

(continued to next slide)

Page Elements [app14]: Vega-Lite Chart *(2)*

```
        "y": {
            "field": "Miles_per_Gallon",
            "type": "quantitative",
        },
        "color": {"field": "Origin", "type": "nominal"},
        "shape": {"field": "Origin", "type": "nominal"},
        "strokeOpacity": {
            "condition": {"param": "point_selection", "value": 1},
            "value": 0.1,
        },
    },
}

tab1, tab2 = st.tabs(["Streamlit theme", "Vega-Lite native theme"])
```

(continued to next slide)

Page Elements [app14]: Vega-Lite Chart *(3)*

```
with tab1:
    # Streamlit theme is the default. So you can also omit the theme.
    event1 = st.vega_lite_chart(
        source, chart, theme = "streamlit",
        key = "vega_chart1", on_select = "rerun",
        use_container_width = True
    )
    event1
with tab2:
    event2 = st.vega_lite_chart(
        source, chart, theme = None,
        key = "vega_chart2", on_select = "rerun",
        use_container_width = True
    )
    event2
```

Vega-Lite: https://vega.github.io/vega-lite/examples/

Page Elements [app15]: Bokeh Chart *(1)*

```
import streamlit as st
from bokeh.models import ColumnDataSource
from bokeh.plotting import figure
from bokeh.sampledata.commits import data
from bokeh.transform import jitter

DAYS = ['Sun', 'Sat', 'Fri', 'Thu', 'Wed', 'Tue', 'Mon']

source = ColumnDataSource(data)

p = figure(
    width = 800,
    height = 300,
    y_range = DAYS,
    x_axis_type = 'datetime',
    title = "Commits by Time of Day (US/Central) 2012-2016",
)
```

(continued to next slide)

Page Elements [app15]: Bokeh Chart *(2)*

```
p.scatter(
    x = 'time',
    y = jitter('day', width = 0.6, range = p.y_range),
    source = source,
    alpha = 0.3,
)

p.xaxis.formatter.days = ['%Hh']
p.x_range.range_padding = 0
p.ygrid.grid_line_color = None

st.bokeh_chart(p, use_container_width = True)
```

Bokeh: https://docs.bokeh.org/en/2.4.3/docs/gallery.html#standalone-examples

Page Elements [app16]: Graphviz Chart

```
import streamlit as st
import graphviz

graph = graphviz.Digraph()
graph.edge("run", "intr")
graph.edge("intr", "runbl")
graph.edge("runbl", "run")
graph.edge("run", "kernel")
graph.edge("kernel", "zombie")
graph.edge("kernel", "sleep")
graph.edge("kernel", "runmem")
graph.edge("sleep", "swap")
graph.edge("swap", "runswap")
graph.edge("runswap", "new")
graph.edge("runswap", "runmem")
graph.edge("new", "runmem")
graph.edge("sleep", "runmem")
st.graphviz_chart(graph)
```

Page Elements [app17]: Plotly Chart

```
import plotly.express as px
import streamlit as st

df = px.data.iris()
fig = px.scatter(
    df,
    x = "sepal_width", y = "sepal_length",
    color = "species",
    size = "petal_length",
    hover_data = ["petal_width"],
)
tab1, tab2 = st.tabs(["Streamlit theme", "Plotly native theme"])
with tab1:
    st.plotly_chart(fig, theme = "streamlit", use_container_width = True)
with tab2:
    st.plotly_chart(fig, theme = None, use_container_width = True)
```

Plotly: https://plotly.com/python/basic-charts/

Page Elements [app18]: PyDeck Chart *(1)*

```
import streamlit as st
import pydeck
import pandas as pd

capitals = pd.read_csv(
    "https://raw.githubusercontent.com/streamlit/docs/refs/heads"
    "/main/python/api-examples-source/data/capitals.csv",
    header = 0,
    names = ["Capital",
             "State",
             "Abbreviation",
             "Latitude",
             "Longitude",
             "Population"],
)
capitals["size"] = capitals.Population / 10
```

(continued to next slide)

Page Elements [app18]: PyDeck Chart *(2)*

```
point_layer = pydeck.Layer(
    "ScatterplotLayer",
    data = capitals,
    id = "capital-cities",
    get_position = ["Longitude", "Latitude"],
    get_color = "[255, 75, 75]",
    pickable = True,
    auto_highlight = True,
    get_radius = "size",
)

view_state = pydeck.ViewState(
    latitude = 40, longitude = -117, controller = True,
    zoom = 2.4, pitch = 30
)
```

(continued to next slide)

Page Elements [app18]: PyDeck Chart *(3)*

```
chart = pydeck.Deck(
    point_layer,
    initial_view_state = view_state,
    tooltip =
        {"text": "{Capital}, {Abbreviation}\nPopulation: {Population}"},
)

event = st.pydeck_chart(
    chart,
    on_select = "rerun",
    selection_mode = "multi-object",
)

event.selection
```

PyDeck: https://deckgl.readthedocs.io/en/latest/

Page Elements [app19]: PyPlot Chart

```
import streamlit as st
import matplotlib.pyplot as plt
import numpy as np

arr = np.random.normal(1, 1, size=1000)
fig, ax = plt.subplots()
ax.hist(arr, bins = 20)

st.pyplot(fig)
```

Matplotlib.PyPlot: https://matplotlib.org/stable/gallery/index.html

Page Elements [app20]: Widgets

```
import streamlit as st

st.button('Click')
st.checkbox('Check the checkbox')
st.radio('Radio Button',[1,2,3])
st.selectbox('Select', [1,2,3])
st.multiselect('Multiselect', [1,2,3])
st.slider('slide', min_value = 0, max_value = 10)
st.text_input('Enter Username')
st.number_input('Enter a Number')
st.text_area('Enter Text Here!')
st.date_input('Date Input')
st.time_input('Time entry', step = 300)
st.file_uploader('File Uploader')
st.color_picker('Select color')
```

Widgets: https://docs.streamlit.io/develop/api-reference/widgets

Page Elements [app21]: Sidebar

```
import streamlit as st

st.sidebar.button("click")
st.sidebar.checkbox('Check the checkbox')
st.sidebar.radio('Radio Button',[1,2,3])
st.sidebar.selectbox('Select', [1,2,3])
st.sidebar.multiselect('Multiselect', [1,2,3])
st.sidebar.slider('slide', min_value = 0, max_value = 10)

with st.sidebar:
    st.text_input('Enter Username')
    st.number_input('Enter a Number')
    st.text_area('Enter Text Here!')
    st.date_input('Date Input')
    st.time_input('Time entry', step = 300)
    st.file_uploader('File Uploader')
    st.color_picker('Select color')
```

Page Elements [app22]: Dialog

```
import streamlit as st

@st.dialog("Cast your vote")
def vote(item):
    st.write(f"Why is {item} your favorite?")
    reason = st.text_input("Because...")
    if st.button("Submit"):
        st.session_state.vote = {"item": item, "reason": reason}
        st.rerun()

if "vote" not in st.session_state:
    st.write("Vote for your favorite")
    if st.button("A"): vote("A")
    if st.button("B"): vote("B")
else:
    f"You voted for {st.session_state.vote['item']} "
    f"because {st.session_state.vote['reason']}"
```

Page Elements [app23]: Callout Messages

```
import streamlit as st

st.error('Error Message', icon = "🚨")
st.warning('Warning Message', icon = "⚠️")
st.info('Info Message', icon = "ℹ️")
st.success('Success Message', icon = "☑️")
st.exception("IndexError('list out of index')")
st.balloons()
```

Page Elements [app24]: Progress

```
import streamlit as st
import time

done = st.empty()
progress_text = "Operation in progress. Please wait."
my_bar = st.progress(0, progress_text)
for percent_complete in range(100):
    time.sleep(0.05)
    my_bar.progress(percent_complete + 1, progress_text)
time.sleep(1)
my_bar.empty()

with st.spinner('Wait for it...'):
    time.sleep(5)
done.write("Done!")

st.button("Rerun")
```

Page Elements [app25]: Status

```
import streamlit as st
import time

with st.status("Data Retrieval", expanded = True) as status:
    st.write("Searching for data...")
    time.sleep(2)
    st.write("Found URL.")
    time.sleep(1)
    st.write("Downloading data...")
    time.sleep(2)
    status.update(
        label = "Download complete!",
        state = "complete",
        expanded = False
    )

st.button("Rerun")
```

App Logic [app26]: Multipage App *(1)*

```
import streamlit as st

def page1():
    st.write(st.session_state.foo)

def page2():
    st.write(st.session_state.bar)

# Widgets shared by all the pages
st.sidebar.selectbox("Foo", ["A", "B", "C"], key = "foo")
st.sidebar.checkbox("Bar", key = "bar")

pg = st.navigation([st.Page(page1), st.Page(page2)])
pg.run()
```

App Logic [app27]: Multipage App *(2)*

```
import streamlit as st

st.title("Multi-Page Application")

pages = {
    "Your account": [
        st.Page("create_account.py", title = "Create your account"),
        st.Page("manage_account.py", title = "Manage your account"),
    ],
    "Resources": [
        st.Page("learn_us.py", title = "Learn about us"),
        st.Page("try_out.py", title = "Try it out"),
    ],
}

pg = st.navigation(pages)
pg.run()
```

App Logic [app28]: Form

```
import streamlit as st

with st.form("my_form"):
    st.write("Inside the form")
    slider_val = st.slider("Form slider")
    checkbox_val = st.checkbox("Form checkbox")

    # Every form must have a submit button.
    submitted = st.form_submit_button("Submit")

st.write("Outside the form")
if submitted:
    with st.container(border = True):
        st.write("Slider:", slider_val)
        st.write("Checkbox:", checkbox_val)
```

App Logic [app29]: Fragment

```
import streamlit as st

if "clicks" not in st.session_state:
    st.session_state.clicks = 0

@st.fragment
def count_to_five():
    if st.button("Plus one!"):
        st.session_state.clicks += 1
        if st.session_state.clicks % 5 == 0:
            st.rerun()
    return

count_to_five()
st.header(f"Multiples of five clicks: {st.session_state.clicks // 5}")
if st.button("Check click count"):
    st.toast(f"## Total clicks: {st.session_state.clicks}")
```

App Logic [app30]: Connection to DB *(1)*

```
import streamlit as st
import pymongo
import pandas as pd

@st.cache_resource
def init_connection():
    return pymongo.MongoClient(**st.secrets["mongo"])

client = init_connection()

@st.cache_data(ttl = 600)
def get_data():
    items = client.fact.restaurants.find()
    items = list(items) # make hashable for st.cache_data
    return items
```

(continued to next slide)

App Logic [app30]: Connection to DB *(2)*

```
df = pd.DataFrame(get_data())
df['last_grade'] = df['grades'].map(lambda x: x[0]['grade’])
df = df.drop(['grades'], axis = 1)

df
```

References

- Streamlit documentation,
  URI: https://docs.streamlit.io
- FACT repository,
  URI: https://github.com/hnovpalit/fact.git