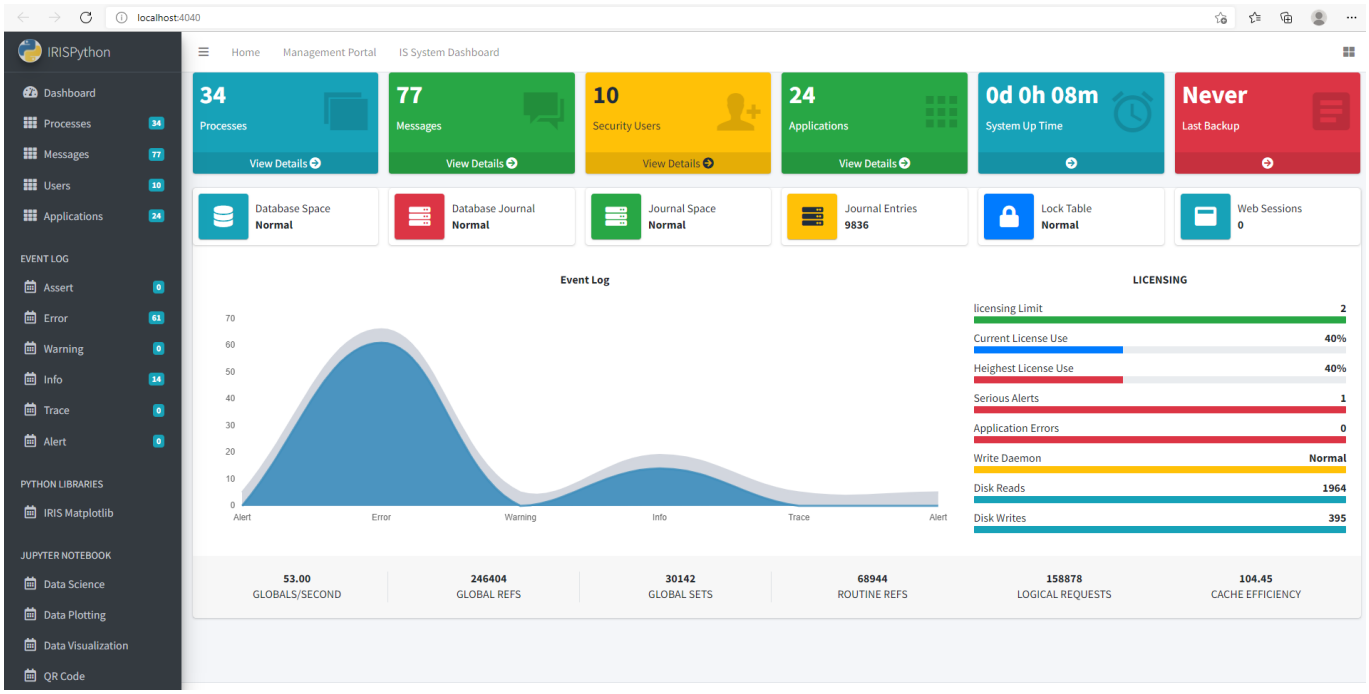
Artículo
Muhammad Waseem · 10 mar, 2022 Lectura de 5 min

Open Exchange

Creación de un cuadro de mando responsive de IRIS con Python Flask Web Framework

¡Hola Comunidad!

Esta publicación es una introducción a mi aplicación iris-python-apps, disponible en Open Exchange y creada usando Embedded Python y Python Flask Web Framework. La aplicación muestra algunas de las funcionalidades de Python, como la ciencia de datos, el trazado de datos, la visualización de datos y la generación de códigos QR.



Características

- **Cuadro de mando de IRIS de arranque *responsive***
- **Vista de los detalles del cuadro de mando junto con el registro de eventos de interoperabilidad y los mensajes**
- **Uso del trazado de Python desde IRIS**
- **Uso de Jupyter Notebook**
- **Introducción a la ciencia de datos, trazado de datos y visualización de datos**
- **Generación de código QR desde Python**

Cuadro de mando de IRIS de arranque *responsive*, usando Python embebido

A continuación se muestra el código de la función definida por el usuario getdashboardstats() para obtener datos desde la clase SYS.Stats.Dashboard IRIS usando Python embebido :

```
import iris
// make sure to connect with %SYS namespace by using custom object script function
 iris.cls("Embedded.Utils").SetNameSpace("%SYS")
 ref = iris.cls("SYS.Stats.Dashboard").Sample()
 last_backup = ref.LastBackup
 #check if variable is empty
 if not last_backup:
     last_backup = "Never"
 #content data dictionary to store data to be used in HTML page
 content = {
     'ApplicationErrors':ref.ApplicationErrors,
     'CSPSessions':ref.CSPSessions,
     'CacheEfficiency':ref.CacheEfficiency,
     'DatabaseSpace' : ref.DatabaseSpace,
     'DiskReads' : ref.DiskReads,
     'DiskWrites' : ref.DiskWrites,
     'ECPAppServer' : ref.ECPAppServer,
     'ECPAppSrvRate' : ref.ECPAppSrvRate,
     'ECPDataServer' : ref.ECPDataServer,
     'ECPDataSrvRate' : ref.ECPDataSrvRate,
     'GloRefs' : ref.GloRefs,
     'GloRefsPerSec' : ref.GloRefsPerSec,
     'GloSets' : ref.GloSets,
     'JournalEntries' : ref.JournalEntries,
     'JournalSpace' : ref.JournalSpace,
     'JournalStatus' : ref.JournalStatus,
     'LastBackup' : last_backup,
     'LicenseCurrent' : ref.LicenseCurrent,
     'LicenseCurrentPct' : ref.LicenseCurrentPct,
     'LicenseHigh' : ref.LicenseHigh,
     'LicenseHighPct' : ref.LicenseHighPct,
     'LicenseLimit' : ref.LicenseLimit,
     'LicenseType' : ref.LicenseType,
     'LockTable' : ref.LockTable,
     'LogicalReads' : ref.LogicalReads,
     'Processes' : ref.Processes,
     'RouRefs' : ref.RouRefs,
     'SeriousAlerts' : ref.SeriousAlerts,
     'ShadowServer' : ref.ShadowServer,
     'ShadowSource' : ref.ShadowSource,
     'SystemUpTime' : ref.SystemUpTime,
     'WriteDaemon' :  ref.WriteDaemon,
     'tot_pro'   : tot_pro,
     'tot_msg'   : tot_msg,
     'tot_usr'   : tot_usr,
     'tot_apps'  : tot_apps,
     'tot_ev' : tot_ev,
     'tot_ev_assert' : tot_ev_assert,
     'tot_ev_error' : tot_ev_error,
     'tot_ev_warning' : tot_ev_warning,
     'tot_ev_info' : tot_ev_info,
     'tot_ev_trace' : tot_ev_trace,
     'tot_ev_alert' : tot_ev_alert
     }
 return content
```

A continuación se muestra el código principal de Python (app.py) para llamar a la función definida por el usuario

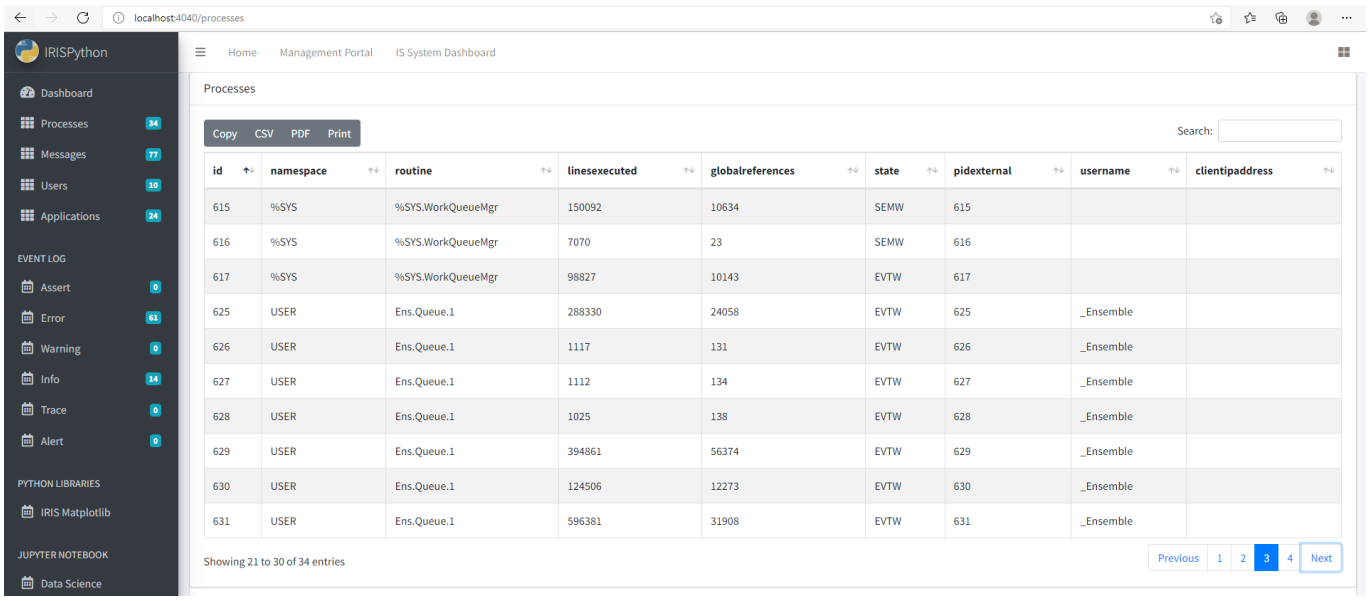
getdashboardstats() para obtener detalles, representar la página index.html y pasarle datos de contenido:

```
@app.route("/")
 def index():
    #get dashboard data in dictionary variable
    content = util.get_dashboard_stats()
    return render_template('index.html', content = content)
```

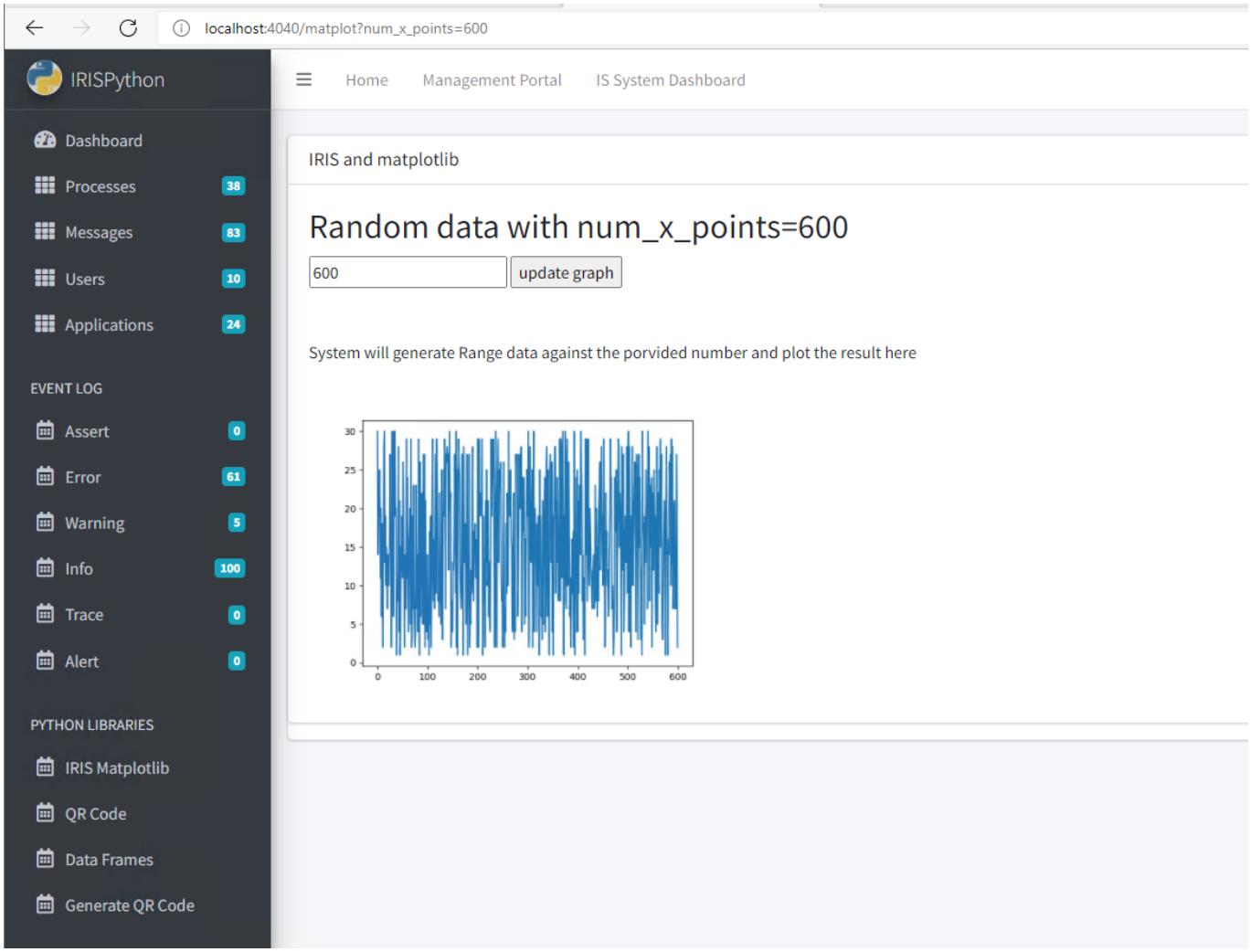
index.html usa la variable del diccionario de contenido para mostrar datos relacionados. Por ejemplo, {{ content.CSPSessions }} se utiliza para mostrar las sesiones CSP:

```
<div class="row">
          <div class="col-lg-2 col-6">
            <!-- small box -->
            <div class="small-box bg-info">
              <div class="inner">
                <h3>{{ content.CSPSessions }}</h3>
                <p>CSP Sessions</p>
              </div>
              <div class="icon">
                <i class="ion ion-ios-photos"></i>
              </div>
              <a href="/processes" class="small-box-
footer">View Details <i class="fas fa-arrow-circle-right"></i></a>
            </div>
 </div>
```

Desde el cuadro de mando también podemos ver los detalles de los procesos en ejecución, los mensajes, los usuarios de seguridad, las aplicaciones y los registros de eventos, mediante el uso de la funcionalidad iris.sql.exec() .



La aplicación también muestra el trazado en HTML mediante el uso de la biblioteca python matplotlib:



A continuación se muestra el código de Python para mostrar el trazado en html:

```
app.route("/matplot")
def matplot():
    #Returns html with the img tag for your plot.
    content = util.get_sidebar_stats()
    num_x_points = int(request.args.get("num_x_points", 50))
    return render_template('matplot.html', content = content, num_x_points = num_x_po
ints)
@app.route("/matplot-as-image-<int:num_x_points>.png")
def plot_png(num_x_points=50):
    # renders the plot on the fly.
    fig = Figure()
    axis = fig.add_subplot(1, 1, 1)
    x_points = range(num_x_points)
    axis.plot(x_points, [random.randint(1, 30) for x in x_points])
    output = io.BytesIO()
    FigureCanvasAgg(fig).print_png(output)
    return Response(output.getvalue(), mimetype="image/png")
```

Introducción a la ciencia de datos, trazado de datos, visualización de datos y generación de QR mediante el uso de Jupyter Notebook

localhost:8888/nbconvert/html/IRISPyDataScience.ipynb?download=false

Data Science by using pandas liabrary

In [2]:
```
import pandas as pd
import numpy as np
import matplotlib as mpl
```

In [2]:
```
df = pd.DataFrame([[38.0, 2.0, 18.0, 22.0, 21, np.nan],[19, 439, 6, 452, 226,232]],
                  index=pd.Index(['Tumour (Positive)', 'Non-Tumour (Negative)'], name='Actual Label:'),
                  columns=pd.MultiIndex.from_product([['Decision Tree', 'Regression', 'Random'],['Tumour', 'Non-Tumour']], names=['Model:', 'Predicted:']))
df.style
```

Out[2]:

Model:	Decision Tree		Regression		Random	
Predicted:	Tumour	Non-Tumour	Tumour	Non-Tumour	Tumour	Non-Tumour
Actual Label:						
Tumour (Positive)	38.000000	2.000000	18.000000	22.000000	21	nan
Non-Tumour (Negative)	19.000000	439.000000	6.000000	452.000000	226	232.000000

In [3]:
```
weather_df = pd.DataFrame(np.random.rand(10,2)*5,
                          index=pd.date_range(start="2021-01-01", periods=10),
                          columns=["Tokyo", "Beijing"])

def rain_condition(v):
    if v < 1.75:
        return "Dry"
    elif v < 2.75:
        return "Rain"
    return "Heavy Rain"

def make_pretty(styler):
    styler.set_caption("Weather Conditions")
    styler.format(rain_condition)
    styler.format_index(lambda v: v.strftime("%A"))
    styler.background_gradient(axis=None, vmin=1, vmax=5, cmap="YlGnBu")
    return styler

weather_df
```

Out[3]:

	Tokyo	Beijing
2021-01-01	3.621969	4.508378
2021-01-02	0.109078	3.062590
2021-01-03	0.493561	1.459806
2021-01-04	3.307517	0.808751
2021-01-05	3.709100	2.608226
2021-01-06	4.165616	0.519589
2021-01-07	4.695969	1.588812
2021-01-08	4.540735	0.477025
2021-01-09	3.149028	3.671971
2021-01-10	3.041754	0.944977

In [4]:
```
weather_df.loc["2021-01-04":"2021-01-08"].style.pipe(make_pretty)
```

Out[4]:

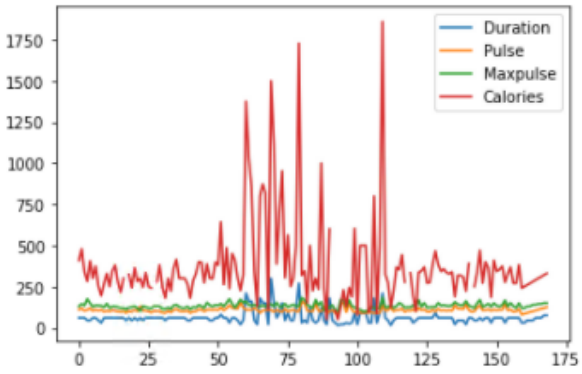
Weather Conditions

	Tokyo	Beijing
Monday	Heavy Rain	Dry
Tuesday	Heavy Rain	Rain
Wednesday	Heavy Rain	Dry
Thursday	Heavy Rain	Dry

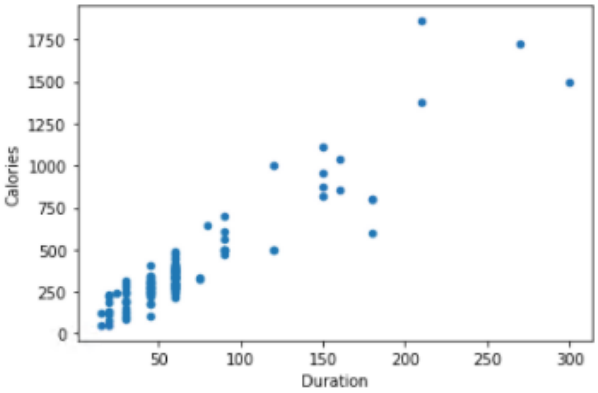
localhost:8888/nbconvert/html/IRISPyMatplotlib.ipynb?download=false

Data plotting by using matplotlib Library

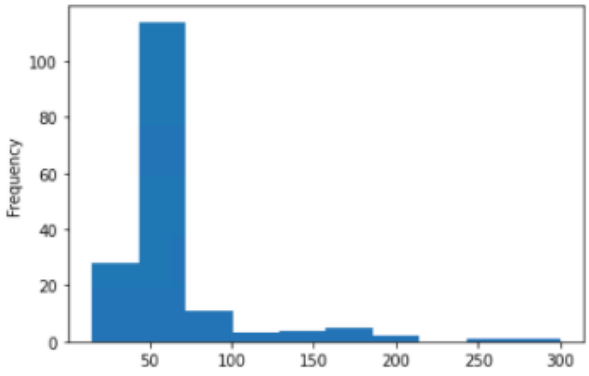
In [11]:
```
%matplotlib inline
import sys
import matplotlib
matplotlib.use('Agg')

import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv('/opt/irisapp/misc/data.csv')

df.plot()

plt.show()
```

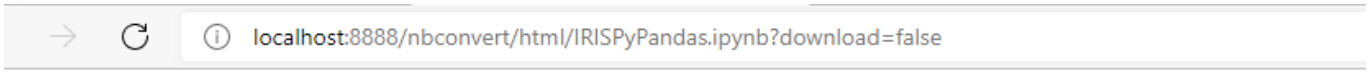


In [13]:
```
df.plot(kind = 'scatter', x = 'Duration', y = 'Calories')
plt.show()
```



In [14]:
```
df["Duration"].plot(kind = 'hist')
plt.show()
```





IRIS Data Visualization by using python pandas liabrary

In [6]:
```
import pandas as pd
pd.options.display.max_rows = 9

import iris
statement = iris.sql.exec('SELECT * FROM Titanic_Table.Passenger')
df = statement.dataframe()
print(df.info())
df
df.duplicated()
df.corr()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   id        891 non-null    int64
 1   survived  891 non-null    int64
 2   pclass    891 non-null    int64
 3   name      891 non-null    object
 4   sex       891 non-null    object
 5   age       891 non-null    int64
 6   sibsp     891 non-null    int64
 7   parch     891 non-null    int64
 8   ticket    891 non-null    object
 9   fare      891 non-null    float64
 10  cabin     891 non-null    object
 11  embarked  891 non-null    object
dtypes: float64(1), int64(6), object(5)
memory usage: 83.7+ KB
None
```

Out[6]:

	id	survived	pclass	age	sibsp	parch	fare
id	1.000000	-0.005007	-0.035144	0.038220	-0.057527	-0.001652	0.012658
survived	-0.005007	1.000000	-0.338481	0.010508	-0.035322	0.081629	0.257307
pclass	-0.035144	-0.338481	1.000000	-0.361566	0.083081	0.018443	-0.549500
age	0.038220	0.010508	-0.361566	1.000000	-0.184593	-0.049061	0.135663
sibsp	-0.057527	-0.035322	0.083081	-0.184593	1.000000	0.414838	0.159651
parch	-0.001652	0.081629	0.018443	-0.049061	0.414838	1.000000	0.216225
fare	0.012658	0.257307	-0.549500	0.135663	0.159651	0.216225	1.000000

Generación de código QR con Python

¡Gracias!

#Bases de datos #CSS #Embedded Python #Paneles de control #Python #Visualización #InterSystems IRIS for Health
Ir a la aplicación en InterSystems Open Exchange

URL de fuente:https://es.community.intersystems.com/post/creaci%C3%B3n-de-un-cuadro-de-mando-responsive-de-iris-con-python-flask-web-framework