



**Tecnológico  
de Monterrey**

# **Analítica de datos y herramientas de inteligencia artificial II**

## **UF6. Desarrollo Web para Apps de Analítica**

### **UF6.2. Frameworks web con Python – Dash Parte4 – Layouts y Despliegue Web**

Profr. Gustavo Romero

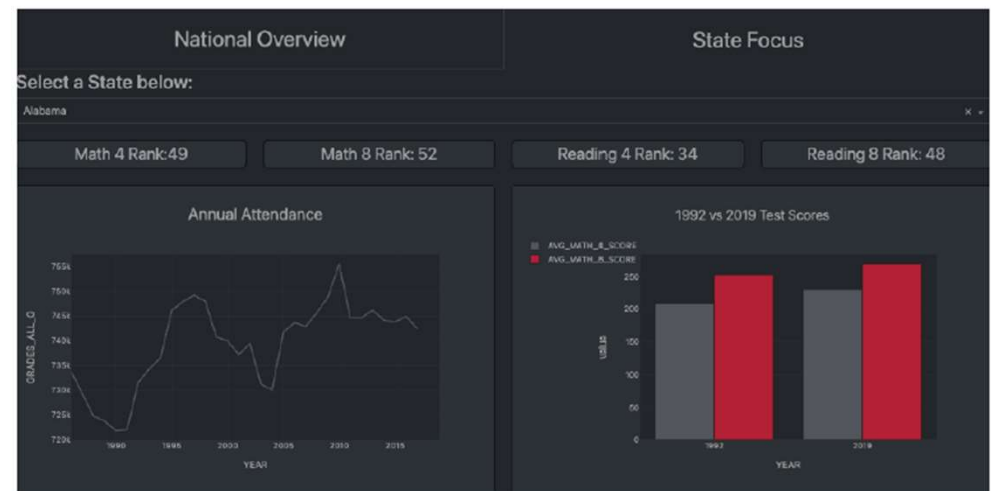
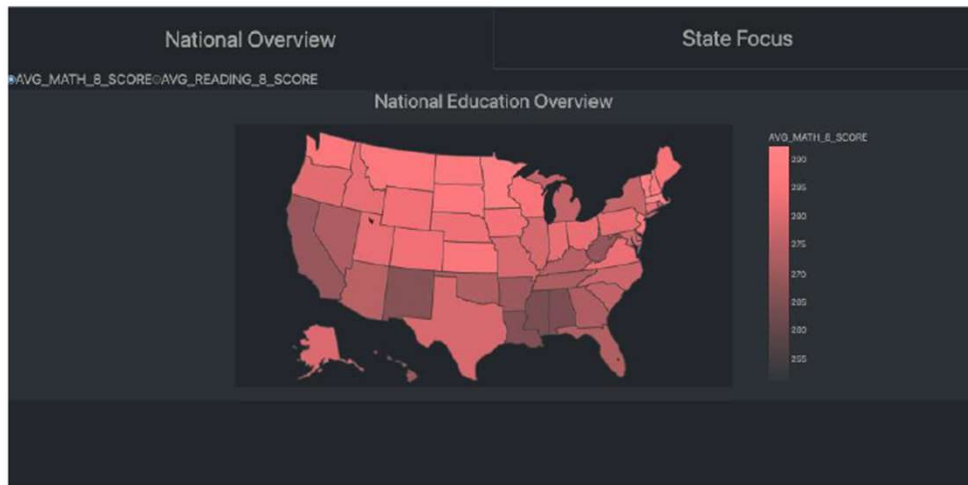
## Layouts

- Los Dashboards son agrupaciones de visualizaciones que permiten apoyar a entender la data. El añadir filtros e interactividad permite al usuario explorar la data para proveerle perspectivas.
- Se crean Dashboard layouts con HTML, markdown, y *Dash Bootstrap Components (DBC)*.
- *DBC permiten crear de manera sencilla layouts con multiples pestañas o páginas matriciales.*
- *El foco de un layout es proveer cohesión a las visualizaciones y elementos interactivos.*
- **Los “temas”** permiten de manera más simple aún aplicar estilos predefinidos a los dashboards, mismos que aún es posible aplicarles personalizaciones a components individuales de ser requerido

# Layouts

Un adecuado layout de dashboard agrega cohesión a sus visualizaciones e interactividad, enfatizando atención a métricas clave y guiando al usuario a una progression lógica.

Se sugiere diseñar un layout de dashboard como una “pirámide invertida”, las métricas y visualizaciones más importantes deben venir primero, seguidas por data de apoyo o visualizaciones más granulares



# Layouts

- Dash usa **HTML** para diseñar el front-end de la aplicación, especificando a través del modulo **html** los diversos components visuals y asignándolos al layout de la app
- Estos son los principales components HTML:

Component	Description
html.Div()	A web page section (you can use multiple Divs to create sections with different styles)
html.H1(), H2(), ..., H6()	Different sized headers used to denote hierarchy or importance (more so than size itself)
html.P()	A paragraph, or generic body text, often smaller than and placed immediately below a header
html.Span()	Inline containers used to apply different colors or styles to text within headers or paragraphs

```
app.layout = html.Div([
    html.H1("This is a Header"),
    html.H2("This is a Header"),
    html.H3("This is a Header"),
    html.H4("This is a Header"),
    html.H5("This is a Header"),
    html.H6("This is a Header"),
    html.P([
        "This is a ",
        html.Span("Paragraph", style={"color": "red"}),
        html.Span(", or body text.")
    ])
])
```



**This is a Header**

**This is a Header**

**This is a Header**

**This is a Header**

**This is a Header**

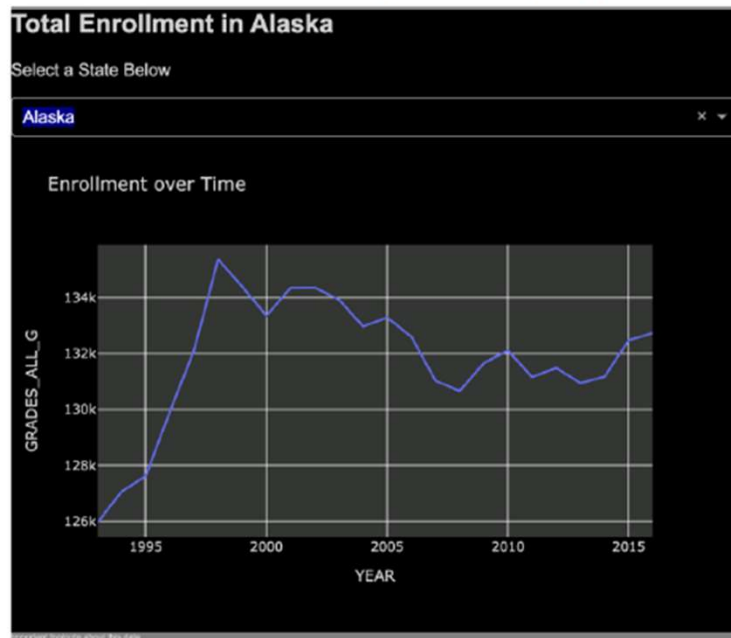
**This is a Header**

This is a **Paragraph**, or body text.

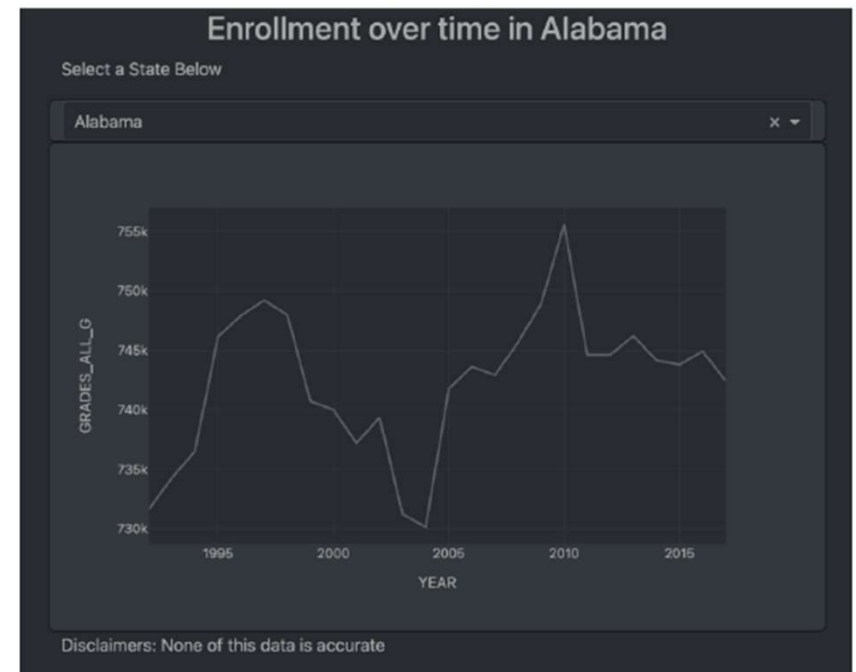
# Layouts

- La librería Dash Bootstrap Components (DBC) ofrece opciones para diseñar apps con pocas líneas de Código, con estilos consistentes, opciones preconstruidas para propiedades de components y un framework matricial.

*Individual component styling*



*Dash Bootstrap Components theme*



# Layouts

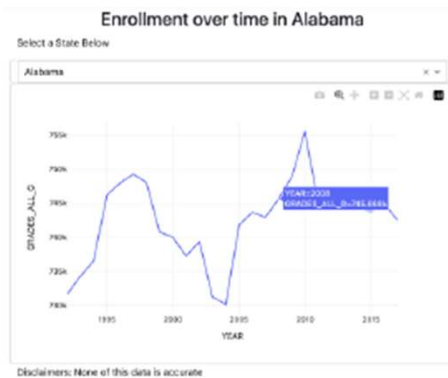
- Estos son los **Dash Bootstrap Components** básicos:

Component	Description
<code>dbc.themes</code>	Pre-built CSS style sheets that apply cohesive formatting to your application
<code>dbc.Container()</code>	The DBC equivalent of a Div that acts as a style wrapper for sections of the app layout
<code>dbc.Card()</code>	A specific type of container for components that adds padding & polish around them
<code>dbc.Row()</code>	Represents a horizontal row inside a <code>dbc.Container</code> (or <code>html.Div</code> )
<code>dbc.Col()</code>	Represents a vertical column inside a <code>dbc.Row</code>
<code>dcc.Tabs()</code>	Creates different tabs for users to navigate between

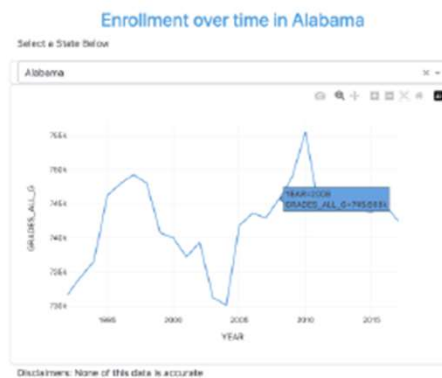
# Layouts

- Existen 26 temas disponibles en la librería Dash Bootstrap Components, y se invocan al crear la app:  
`Dash(__name__, external_stylesheets=[dbc.themes.THEME_NAME])`
- Existe un link para poder explorar los temas disponibles y poder visualizarlos para elegir alguno  
<https://dash-bootstrap-components.opensource.faculty.ai/docs/themes/explorer/>

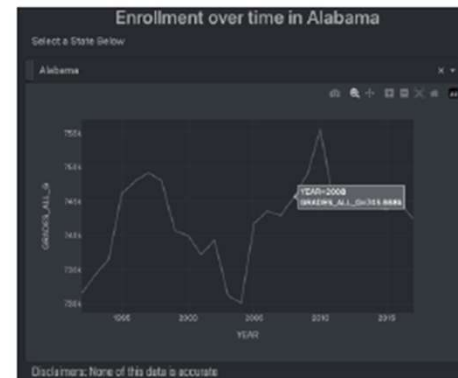
**Bootstrap**



**Cerulean**



**Slate**



**Quartz**



# Layouts

- Es posible aplicar un tema a las figuras o gráficos utilizando la librería `dash_bootstrap_templates`

`load_figure_template("THEME_NAME")`

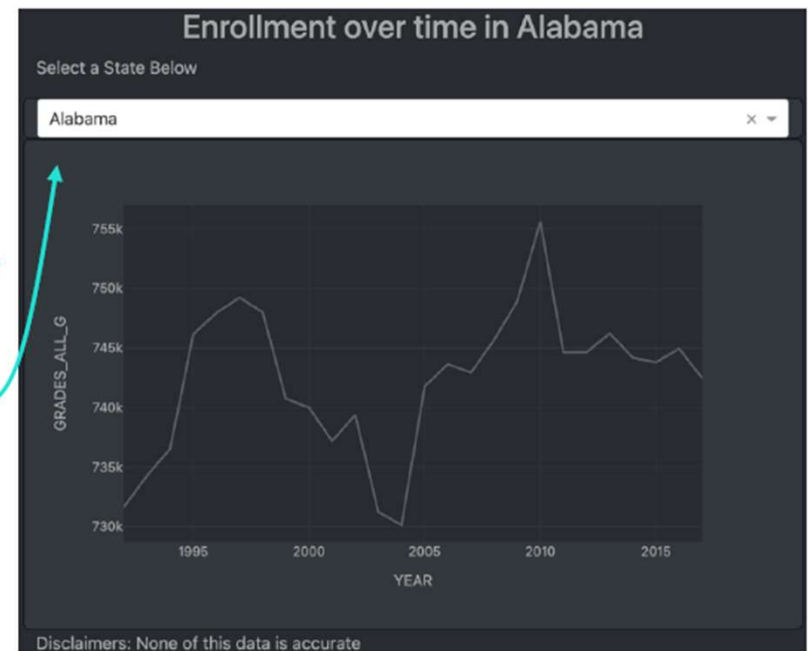
1) *Install the library*

```
from os import sys
!pip install --prefix {sys.prefix} dash_bootstrap_templates
```

2) *Specify the desired theme*

```
from dash_bootstrap_templates import load_figure_template
app = JupyterDash(__name__, external_stylesheets=[dbc.themes.SLATE])
load_figure_template("SLATE")
```

Sólo falta lidiar con algunos  
elementos independientes





# Layouts

- Es posible aplicar temas a componentes DCC utilizando un link especial para importar estilos
- También es necesario especificar **className="dbc"** en el componente dcc

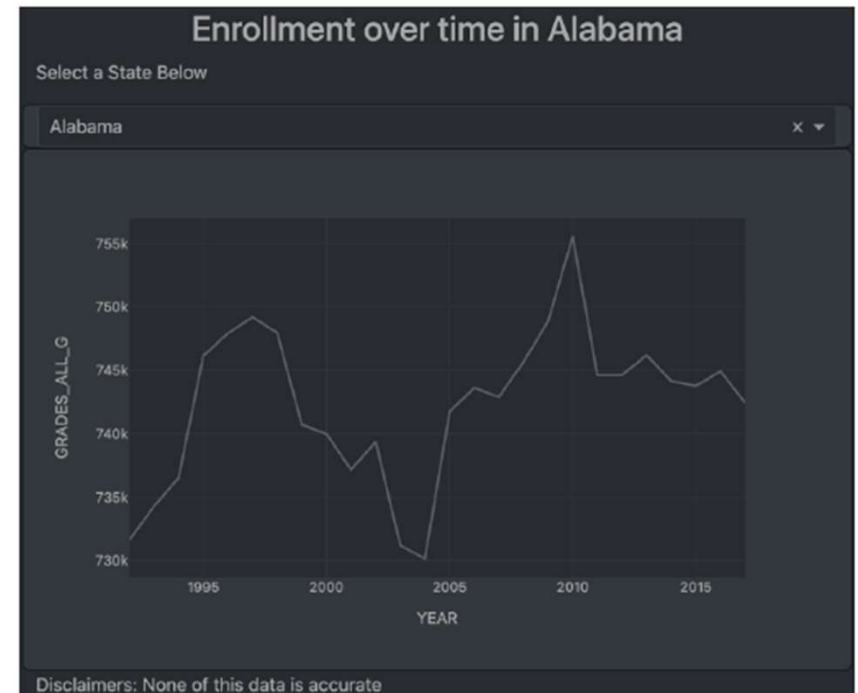
```
dbc_css = "https://cdn.jsdelivr.net/gh/AnnMarieW/dash-bootstrap-templates/dbc.min.css"

app = JupyterDash(__name__, external_stylesheets=[dbc.themes.SLATE, dbc_css])
load_figure_template("SLATE")

app.layout = dbc.Container(
    children=[
        dbc.Row(
            html.H2(
                id="Header Text",
                style={
                    "textAlign": "center",
                },
            ),
            html.P("Select a State Below", id='instructions'),
            dbc.Row(
                dbc.Card(
                    dcc.Dropdown(
                        options=["Alabama", "Alaska", "Arkansas"],
                        value="Alabama",
                        id="State Dropdown",
                        className="dbc"
                    ),
                ),
                dbc.Row(
                    dbc.Card(dcc.Graph(id="Revenue Line")),
                ),
            ),
            html.P("Disclaimers: None of this data is accurate")
        ],
    )
)
```

*Imports external CSS  
that applies to specified  
dcc components*

*Applies theme to the component*



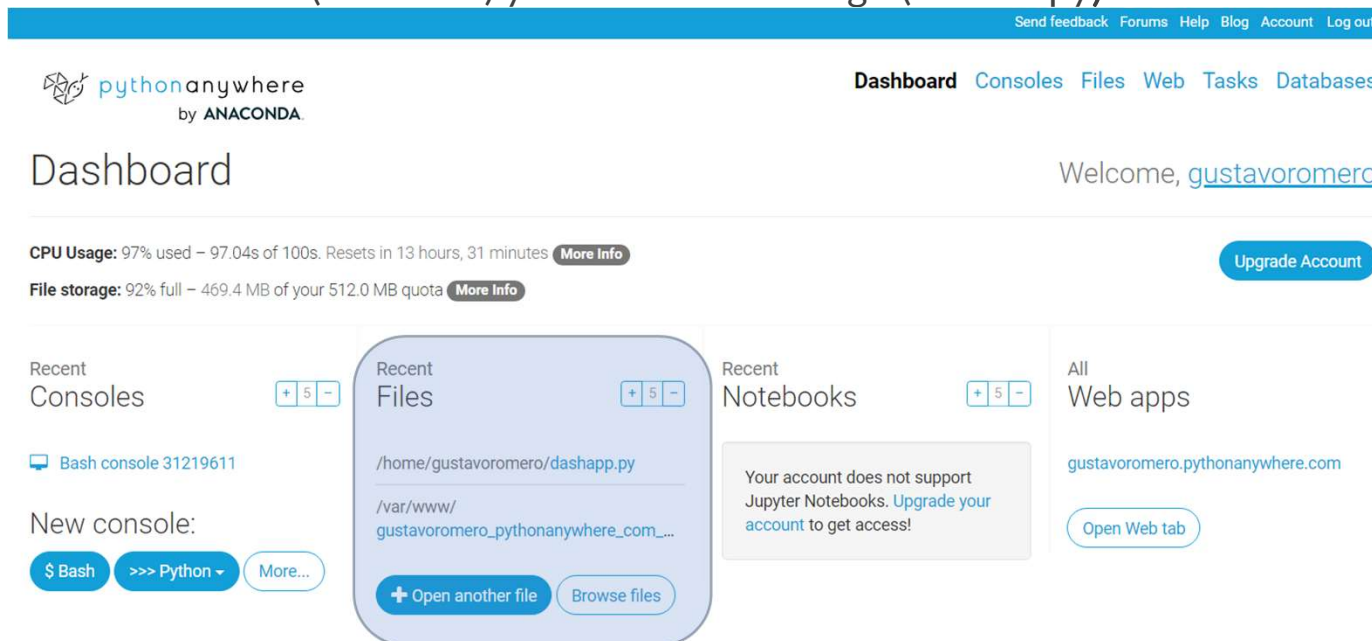
## Preparando para Despliegue web

- Opcional: Agregar autenticación básica
  - 1) Instalar modulo dash-auth → `!conda install --yes --prefix {sys.prefix} dash-auth`
  - 2) Importar módulo dash\_auth → `import dash_auth`
  - 3) Crear pares de usuarios y contraseñas →  
`USERNAME_PASSWORD_PAIRS = [['usuario1', 'pass1'],`  
`['usuario2', 'pass2']]`
  - 4) Agregar autorización después de la creación de la app →  
`auth = dash_auth.BasicAuth(app, USERNAME_PASSWORD_PAIRS)`

# Despliegue web (Python Anywhere)

- 1) Crear una cuenta gratuita en [pythonanywhere.com/pricing](https://pythonanywhere.com/pricing)
- 2) Una vez registrado, desde el Dashboard añadir archivos de data y app

Subirás archivos de data utilizados (como csv) y el archivo de Código (archivo.py)



The screenshot shows the Python Anywhere dashboard for a user named gustavoromero. The dashboard includes a top navigation bar with links for Send feedback, Forums, Help, Blog, Account, and Log out. Below the navigation bar, the dashboard is divided into several sections. On the left, there's a 'Recent Consoles' section showing a 'Bash console 31219611'. In the center, there's a 'Recent Files' section showing two files: '/home/gustavoromero/dashapp.py' and '/var/www/gustavoromero\_pythonanywhere\_com\_...'. On the right, there's a 'Recent Notebooks' section with a message stating 'Your account does not support Jupyter Notebooks. Upgrade your account to get access!'. At the bottom right, there's an 'All Web apps' section showing 'gustavoromero.pythonanywhere.com' with an 'Open Web tab' button. The dashboard also features a 'CPU Usage' section showing 97% used and a 'File storage' section showing 92% full. A 'New console' section at the bottom left offers options for '\$ Bash' and '>>> Python'. A 'More...' button is also present.

Send feedback Forums Help Blog Account Log out

pythonanywhere  
by ANACONDA

Dashboard Consoles Files Web Tasks Databases

Welcome, [gustavoromero](#)

CPU Usage: 97% used – 97.04s of 100s. Resets in 13 hours, 31 minutes [More Info](#)

File storage: 92% full – 469.4 MB of your 512.0 MB quota [More Info](#)

Upgrade Account

Recent Consoles [+](#) [5](#) [-](#)

Bash console 31219611

New console:

\$ Bash >>> Python [More...](#)

Recent Files [+](#) [5](#) [-](#)

/home/gustavoromero/dashapp.py

/var/www/gustavoromero\_pythonanywhere\_com\_...

[+ Open another file](#) [Browse files](#)

Recent Notebooks [+](#) [5](#) [-](#)

Your account does not support Jupyter Notebooks. [Upgrade your account](#) to get access!

All Web apps

[gustavoromero.pythonanywhere.com](#)

[Open Web tab](#)

# Despliegue web (Python Anywhere)

- 3) Ir a la pestaña “Web” y añadir una nueva web app



The screenshot shows the Python Anywhere dashboard with the 'Web' tab selected. The 'Add a new web app' button is highlighted. A callout box explains that clicking this button leads to the creation of a PythonAnywhere-hosted web app. The subsequent steps in the wizard are shown with annotations:

- Your web app's domain name:** The default domain `everest1.pythonanywhere.com` is selected, with a note to 'Use the default domain'.
- Select a Python Web framework:** The 'Manual configuration (including virtualenvs)' option is selected, with a note to 'Click “Manual Configuration” and select Python “3.10”'.
- Manual Configuration:** The screen explains that manual configuration involves editing the WSGI configuration file and that a 'Hello World' app will be created. The 'Next >' button is highlighted.

Navigation links at the top include: Dashboard, Consoles, Files, **Web**, Tasks, Databases.

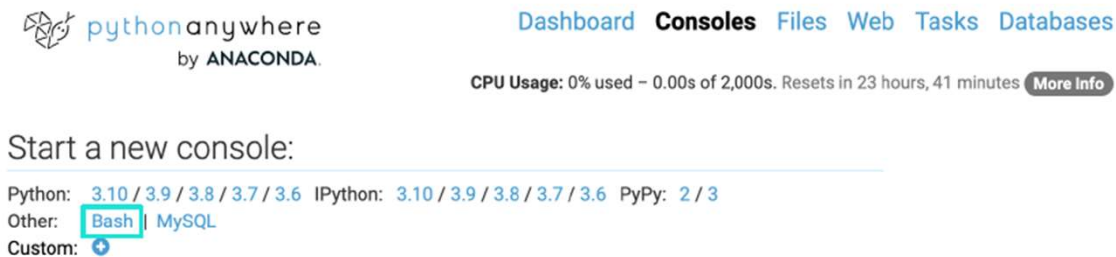
# Despliegue web (Python Anywhere)

- 4) Verificar que la app por default está activa



The screenshot shows the Python Anywhere dashboard for the user 'gustavoromero.pythonanywhere.com'. The 'Web' tab is selected in the top navigation bar. The main content area displays 'Hello, World!' in a large font, with a sub-message stating: 'This is the default welcome page for a [PythonAnywhere](#) hosted web application. Find out more about how to configure your own web application by visiting the [web app setup](#) page'. A green button labeled 'Reload' is visible below the configuration section.

- 5) Ir a pestaña de “Consoles” e iniciar una nueva Consola **Bash**



The screenshot shows the Python Anywhere dashboard with the 'Consoles' tab selected. The 'Start a new console:' section is visible, showing a list of options: 'Python: 3.10 / 3.9 / 3.8 / 3.7 / 3.6', 'IPython: 3.10 / 3.9 / 3.8 / 3.7 / 3.6', and 'PyPy: 2 / 3'. Under the 'Other:' category, 'Bash' is highlighted with a red box, and 'MySQL' is also listed. A 'Custom:' option with a plus icon is at the bottom. The top navigation bar shows 'Consoles' as the active tab.

## Despliegue web (Python Anywhere)

- 6) En la terminal crea un entorno, teclea:
- `mkvirtualenv myvirtualenv --python=/usr/bin/python3.10`
- Es importante que una vez creado asegures tener instalados todos los módulos que requerirás para la ejecución de tu aplicación. Se pueden verificar con: `pip list --local`
- De lo contrario los instalarás. Ej:
  - `pip install dash`
  - `pip install dash_auth`
  - `pip install dash_bootstrap_components`
  - `pip install dash_bootstrap_templates`
  - `pip install pandas`



Bash console 31219611

```
(myvirtualenv) 16:33 ~ $  
(myvirtualenv) 16:57 ~ $ pip list --local  
Package                               Version  
-----  
ansi2html                             1.8.0  
blinker                               1.7.0  
certifi                               2023.11.17  
charset-normalizer                    3.3.2  
click                                 8.1.7  
dash                                  2.14.1  
dash-auth                             2.0.0  
dash-bootstrap-components             1.5.0  
dash-bootstrap-templates              1.1.1
```

## Despliegue web (Python Anywhere)

- 7) En la pestaña “Web” conecta tu app a tu entorno

Virtualenv:

Use a virtualenv to get different versions of flask, django etc from our default system ones. [More info here](#). You need to **Reload your web app** to activate it; NB - will do nothing if the virtualenv does not exist.

[/home/gustavoromero/.virtualenvs/myvirtualenv](#)

[Start a console in this virtualenv](#)

- 8) Edita el archivo de configuración WSGI con el Código siguiente:

Code:

What your site is running.


Source code: [/home/gustavoromero/dashapp.py](#)

[Go to directory](#)

Working directory: [/home/gustavoromero/](#)

[Go to directory](#)

WSGI configuration file: [/var/www/gustavoromero-pythonanywhere\\_com\\_wsgi.py](#)

Python version: 3.10 

```
1 import sys
2
3 project_home = u'/home/gustavoromero/'
4 if project_home not in sys.path:
5     sys.path = [project_home] + sys.path
6
7 from dashapp import app
8 application = app.server
```


Nombre de tu archivo de tu app .py

## Despliegue web (Python Anywhere)

- 9) Ajustes al Código para el despliegue (lo puedes hacer directo desde “Files”):
  - Asegúrate de estar utilizando **Dash** y no **Jupyter\_Dash**
  - Retira la última línea de Código donde ejecutas la app en el servidor local
- 10) Especifica la ruta para tu app en el “Source code” en la pestaña “Web”

Code:

What your site is running.


Source code:	<a href="/home/gustavoromero/dashapp.py">/home/gustavoromero/dashapp.py</a>	<a href="#">➔Go to directory</a>
Working directory:	<a href="/home/gustavoromero/">/home/gustavoromero/</a>	<a href="#">➔Go to directory</a>
WSGI configuration file:	<a href="/var/www/gustavoromero-pythonanywhere_com_wsgi.py">/var/www/gustavoromero_pythonanywhere_com_wsgi.py</a>	
Python version:	3.10 	



# Despliegue web (Python Anywhere)

- 11) Recarga la configuración y visualiza tu app

Send feedback Forums Help Blog Account Log out

 pythonanywhere  
by ANACONDA

[Dashboard](#) [Consoles](#) [Files](#) **Web** [Tasks](#) [Databases](#)

[gustavoromero.pythonanywhere.com](#)

[Add a new web app](#)

Configuration for  
[gustavoromero.pythonanywhere.com](#)

Reload:

[Reload gustavoromero.pythonanywhere.com](#)

← → ↻ ⚠ Not secure | gustavoromero.pythonanywhere.com

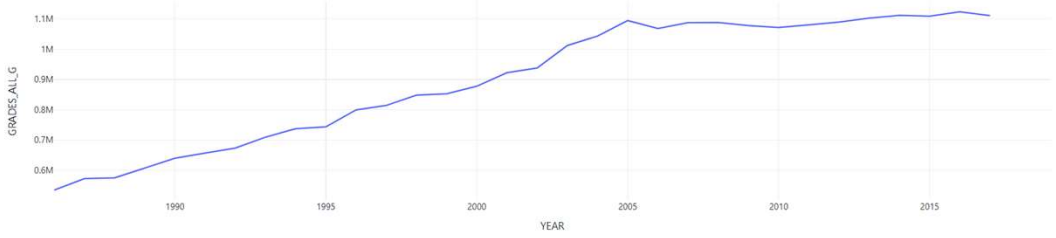
★ Bookmarks Cisco Networking A... Click Subscribe | Co... Ingram Micro - Login Matellus HD Signy - Login Vortice - Vinculació... Comprime PDF - Re... Mesa Matellus Lupita » All Bookmark

Actual Layout Tab Just for illustration

Education Stats for Arizona

ARIZONA x ▾

Annual Attendance



## Despliegue web (Heroku)

- A PARTIR DE NOVIEMBRE DE 2022 HEROKU NO DISPONE DE ALGUNA OPCION GRATUITA.

SE DEJA COMO REFERENCIA UNA BREVE GUIA PARA SU USABILIDAD

## Despliegue web (Heroku)

- 1) Instalar Heroku y Git
  - Heroku es una Plataforma Cloud que permite el despliegue de apps a web, con GIT se contará con un Sistema de control de versionado para hacer push de nuestro entorno de Desarrollo a la version productiva en Heroku.
  - a) Crear una cuenta en Heroku y loguearse: <https://signup.heroku.com/account>
  - b) Seleccionar Python y crear un app. En Set Up descargar “Heroku Command Line Interface” e instalarla

- 2) Instalar entorno virtual → en la terminal **pip install virtualenv**

- 3) Crear una carpeta de Desarrollo que aloje la copia e inicializa Git →

```
C:\>mkdir my_dash_app
C:\>cd my_dash_app
C:\my_dash_app>git init
Initialized empty Git repository in C:/my_dash_app/.git/
```

- 4) Crea, Activa y conforma el virtualenv

```
C:\my_dash_app>python -m virtualenv venv
```

```
C:\my_dash_app>.\venv\Scripts\activate
```

Nota es possible se deba ejecutar si no permite la activación **Set-ExecutionPolicy Unrestricted**

```
(venv) C:\my_dash_app>pip install dash
(venv) C:\my_dash_app>pip install dash-auth
(venv) C:\my_dash_app>pip install dash-renderer
(venv) C:\my_dash_app>pip install dash-core-components
(venv) C:\my_dash_app>pip install dash-html-components
(venv) C:\my_dash_app>pip install gunicorn
```

## Despliegue web (Heroku)

- 5) Agrega los siguientes archivos al Folder de Desarrollo.

app1.py	a Dash application
.gitignore	used by git, identifies files that <i>won't</i> be pushed to production
Procfile	used for deployment
requirements.txt	describes your Python dependencies, can be created automatically

- a) copia el archivo con le Código del app que deseas desplegar y añade el Código **server = app.server** después de la línea de creación de app. El archivo se llama **app1.py**
- b) crea los archivos **.gitignore** y **Procfile** con el siguiente contenido:

### **.gitignore**

```
venv  
*.pyc  
.DS_Store  
.env
```

### **Procfile**

```
web: gunicorn app1:server
```

- c) Generar el archivo requirements.txt corriendo desde terminal el comando:

```
(venv) C:\my_dash_app>pip freeze > requirements.txt
```

## Despliegue web (Heroku)

- 6) Loguearse en la cuenta de Heroku desde terminal

```
(venv) C:\my_dash_app>heroku login
Enter your Heroku credentials:
Email: my.name@somewhere.com
Password: *****
Logged in as my.name@somewhere.com
```

- 7) Inicializar Heroku, añadir archivos a Git y desplegar

```
(venv) C:\my_dash_app>heroku create my-dash-app
You have to change my-dash-app to a unique name. The name must start with a letter
and can only contain lowercase letters, numbers, and dashes.
(venv) C:\my_dash_app>git add .
Note the period at the end. This adds all files to git (except those listed in .gitignore)
(venv) C:\my_dash_app>git commit -m "Initial launch"
Every git commit should include a brief descriptive comment. Depending on your operating system, this
comment may require double-quotes (not single-quotes).
(venv) C:\my_dash_app>git push heroku master
This deploys your current code to Heroku. The first time you push may take awhile as it has to set up Python and
all your dependencies on the remote server.
(venv) C:\my_dash_app>heroku ps:scale web=1
Scaling dynos... done, now running web at 1:Free
This runs the app with a 1 heroku "dyno"
```

- 8) Visualiza la app → <https://my-dash-app.herokuapp.com>

## Despliegue web (Heroku)

- 9) Si haces cambios a tu app:

If installing a new package:

```
$ pip install newdependency  
$ pip freeze > requirements.txt
```

If updating an existing package:

```
$ pip install dependency --upgrade  
$ pip freeze > requirements.txt
```

In all cases:

```
$ git status # view the changes (optional)  
$ git add . # add all the changes  
$ git commit -m "a description of the changes"  
$ git push heroku master
```