Casos de uso de Python

Daniel Molina Cabrera

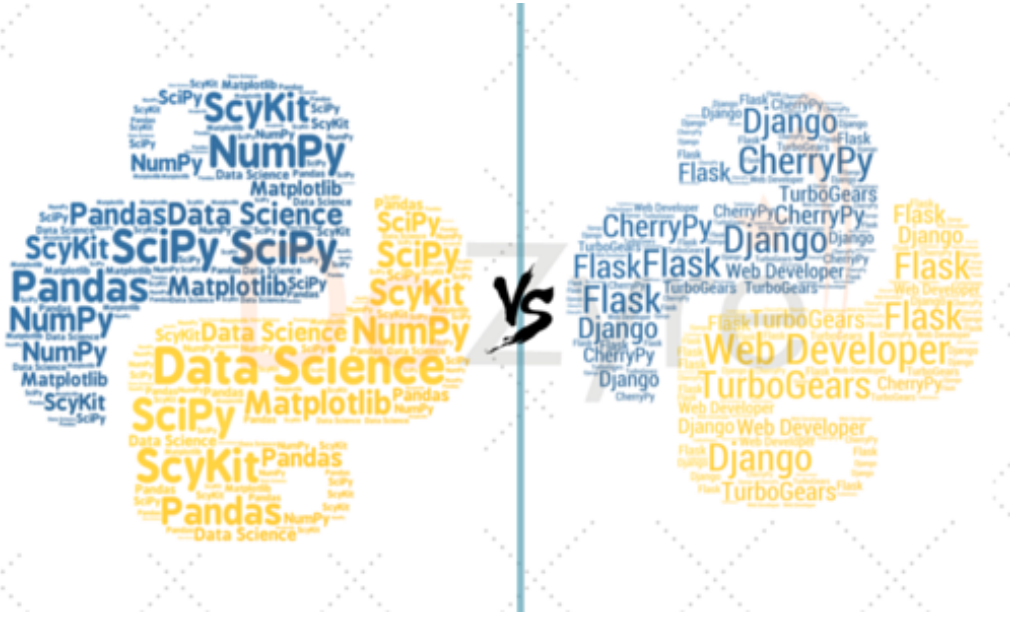
Curso de Python (Abril 2018)

Casos de uso de Python

Qué veremos

- Áreas donde Python aporta.
- Librerías recomendadas.

Áreas que veremos



Ciencia

- Cálculo: Numpy, Pandas.
- Visualización.

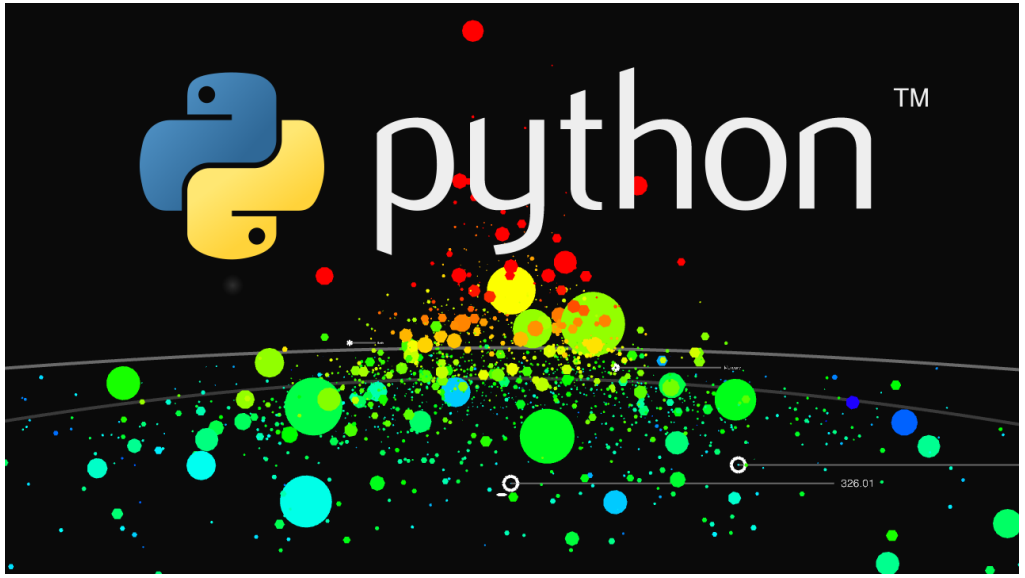
Programación web

- Backend: Django, Flask.

Índice

1. Python para la ciencia
2. Python y la programación web

Python para la ciencia



Python para la ciencia

- Python se usa mucho para la ciencia de datos[a].
- Gran parte de su comunidad son científicos, no informáticos.

[a] http://awahid.net/blog/data-science-with-python-or-java/

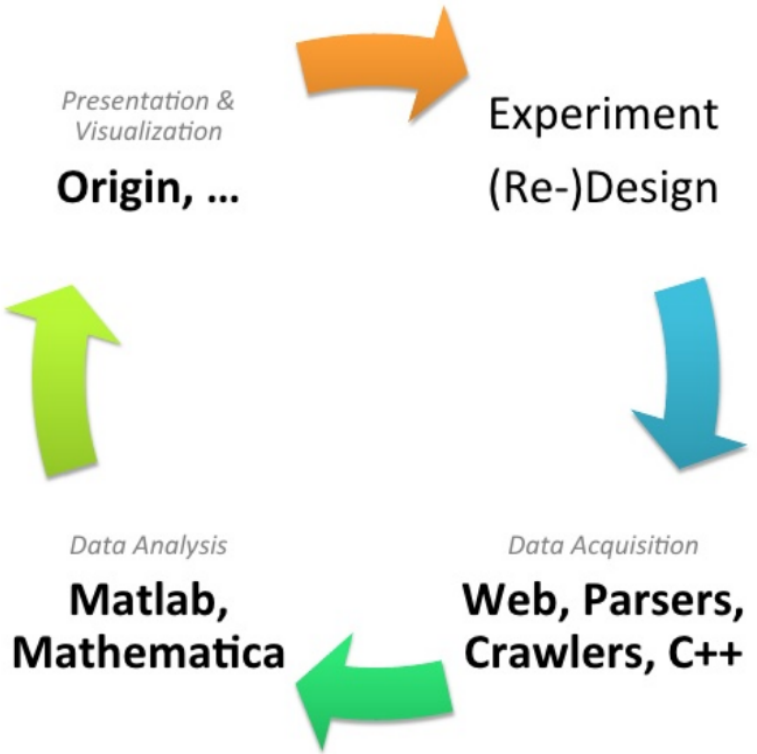
¿Qué aporta Python en la ciencia?

Ventajas

- Lenguaje fácil de usar para no informáticos.
- Comunidad amigable.
- Librerías científicas avanzadas fáciles de usar.
- Entorno desarrollo estable.
- Paralelismo.

Utilidad de Python

Utilidad en proceso científico [1]



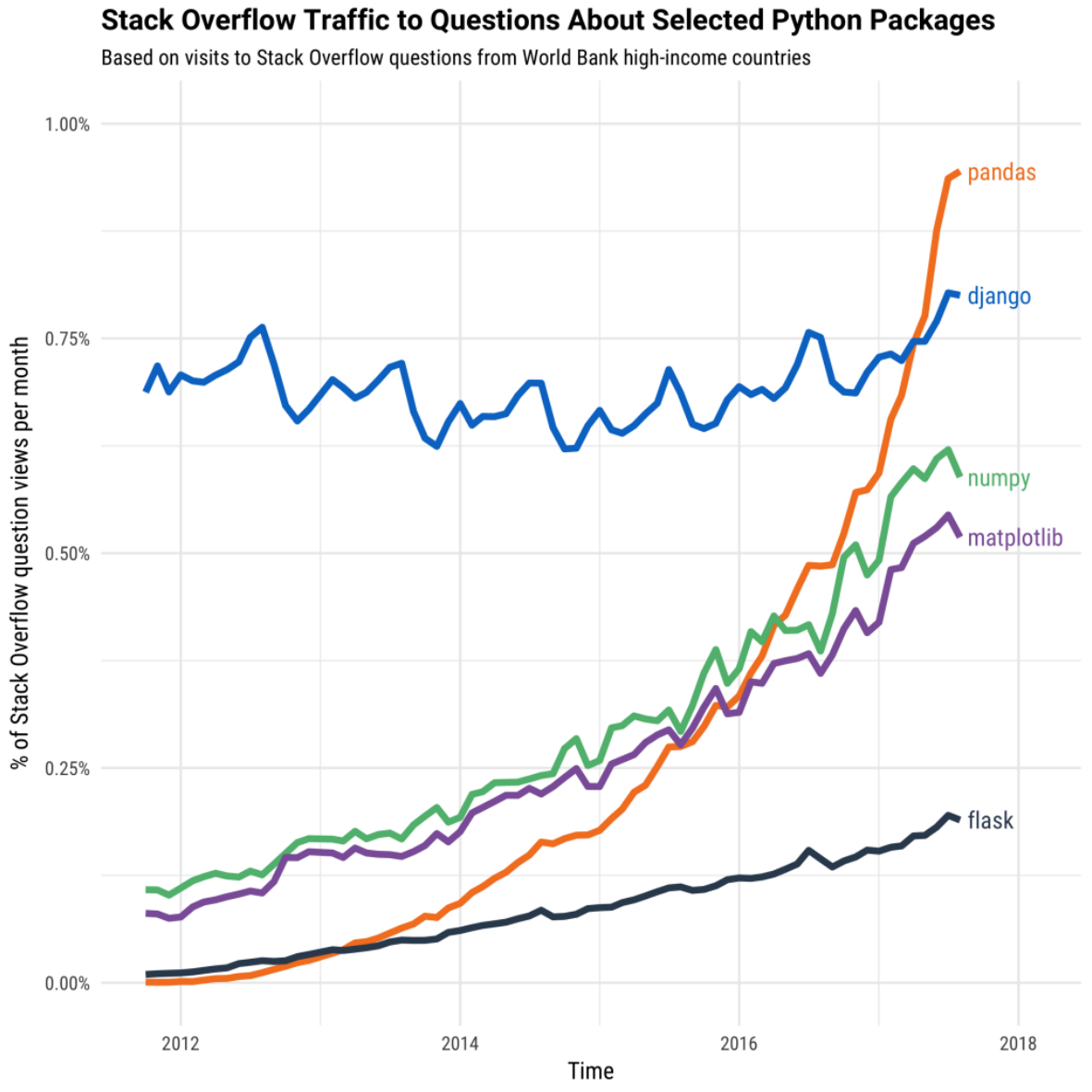
[1] https://www.slideshare.net/marcelcaraciolo/computao-cientfica-com-python-numpy-e-scipy

Utilidad de Python

Utilidad en proceso científico [1]

[1] https://www.slideshare.net/marcelcaraciolo/computao-cientfica-com-python-numpy-e-scipy

Uso de Python



Rendimiento de Python

¿Pero Python no era lento?

Python por defecto sí[a]

Table: Tiempo de un benchmark (s)

Versión	Tiempo (ms)
Python puro	183
Numpy	5.97
Cython normal	7.76
Cython optimizado	2.18
Cython llamando a C	2.22

[a] https://arogozhnikov.github.io/2015/01/06/benchmarks-of-speed-numpy-vs-all.html

Rendimiento de Python

¿Pero Python no era lento?[1]

Table: tiempo matriz de distancias

Versión	Tiempo
Python	9.51 seg
Naive numpy	64.7 ms
Numba	6.72 ms
Cython	6.57 ms
Parakeet	12.3 ms
Cython	6.57 ms

Rendimiento en Python

- Procesar 1 GB de datos de Datos
  - 145.232 filas y 1.936 variables.
- Comparando Python vs Scala (Java)[2]:

Usando Spark

Nodos	Versión	Tiempo
3	Scala	250 s
3	Python	246 s

[2] https://stackoverflow.com/questions/32464122/spark-performance-for-scala-vs-python

Rendimiento en Python

Hay distintas alternativas

Librería numpy Equivalente a Matlab, optimizable con BLAS.

PyPy Intérprete JIT, en migración a Python3.

Cython Python compilado a C (Python + tipos).

Numba Compilación JIT.

Y en paralelo

Paralelismo fácil Clusters, snakemake, Luigi.

Librerías paralelas Dask.

Big Data pyspark.

Librerías en GPU PyCUDA, PyTorch.

Caso de ejemplo: Numpy + Pandas

Numpy

- Librería matricial potente.
- Pandas, leer tablas de datos.

Numpy es rápida y potente

Usando Python

```
def disteuc(xs,ys):
    sum = 0

    for x, y in zip(xs, ys):
        sum += (x-y)*(x-y)

    return sqrt(sum)

%time disteuc(xs,ys)
```

Distancia euclídea con numpy

```
def disteucnp(xs,ys):
    z = (xs-ys)
    return sqrt((z*z).sum())

%time disteucnp(xs,ys)
```

CPU times: user 144 ms, sys: 0 ns, total: 144 ms Wall time: 142 ms

CPU times: user 8 ms, sys: 0 ns, total: 8 ms Wall time: 9.63 ms

https://docs.scipy.org/doc/numpy/user/quickstart.html

Pandas

Permite

- Agrupar datos de distinto tipo.
- Leer y escribir en csv, Excel.
- Filtras y agrupar por ciertos atributos.

Algunos ejemplo

- DataSets de propinas.
- DataSets del Titanic.
- DataSets de nombres por años.

https://pandas.pydata.org/pandas-docs/stable/10min.html

Visualizando [3]

Múltiples librerías

Matplotlib	Librería por defecto, basada en Matlab.
Seaborn	Sobre matplotlib, estilos.
Pandas	Directamente.
Bokeh	Gráficas webs.
Holoviews	Sobre Bokeh, mayor nivel abstracción.
Altair	Enfoque declarativo, web, en desarrollo.

[3] https://bit.ly/2sUHcJu

Visualizando

¿Qué aporta?

- Excel ya me permite hacer gráficas.
- Excel ya gestiona tablas.

Qué ofrece Python

- Obtener datos de fuentes distintas
- Análisis de datos visualmente
- Diagramas interactivos

Qué ofrece Python

Obtener datos de fuentes distintas

- Redes sociales.
- Páginas webs (*web scraping*).
- Otros recursos (Bases de Datos, . . .).

Análisis de datos visualmente

- Interactivo: Notebook.
- Forma declarativa.

Diagramas interactivos

- Explorar datos.
- Panel de control interactivo.

Ejemplo: Caso de estudio

Vamos a ver un ejemplo

Altair Librería en contrucción declarativa.

Objetivo Explorar tiempo de Seattle.

Datos

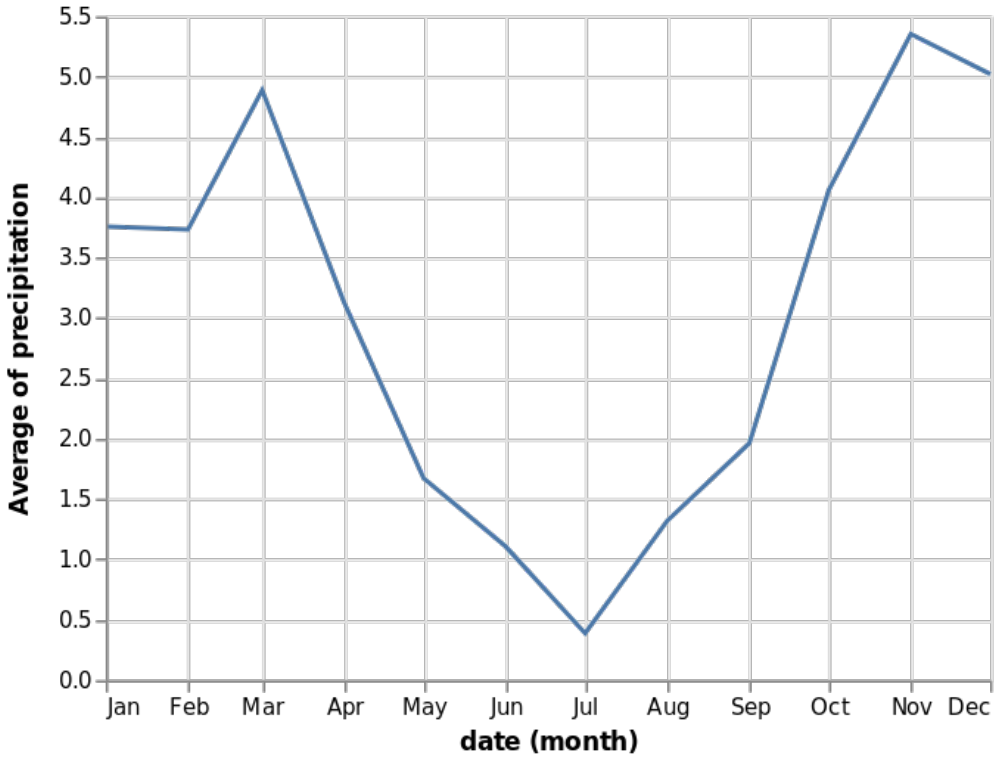
```
df.head()
```

date	precipitation	temp_max	temp_min	wind	weather
2012-01-01	0.0	12.8	5.0	4.7	drizzle
2012-01-02	10.9	10.6	2.8	4.5	rain
2012-01-03	0.8	11.7	7.2	2.3	rain
2012-01-04	20.3	12.2	5.6	4.7	rain
2012-01-05	1.3	8.9	2.8	6.1	rain

Precipitaciones por mes

Código

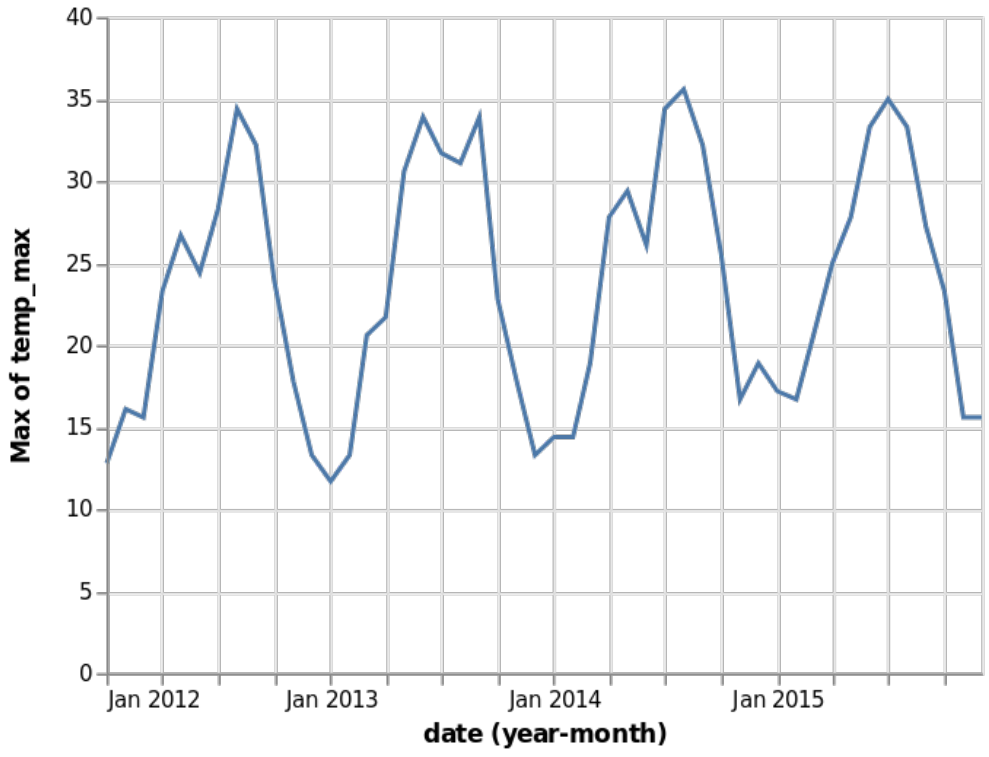
```
alt.Chart(df).mark_line().encode(
    alt.X("date:T", timeUnit="month"),
    alt.Y("average(precipitation)")
)
```



Precipitaciones por año y mes

Código

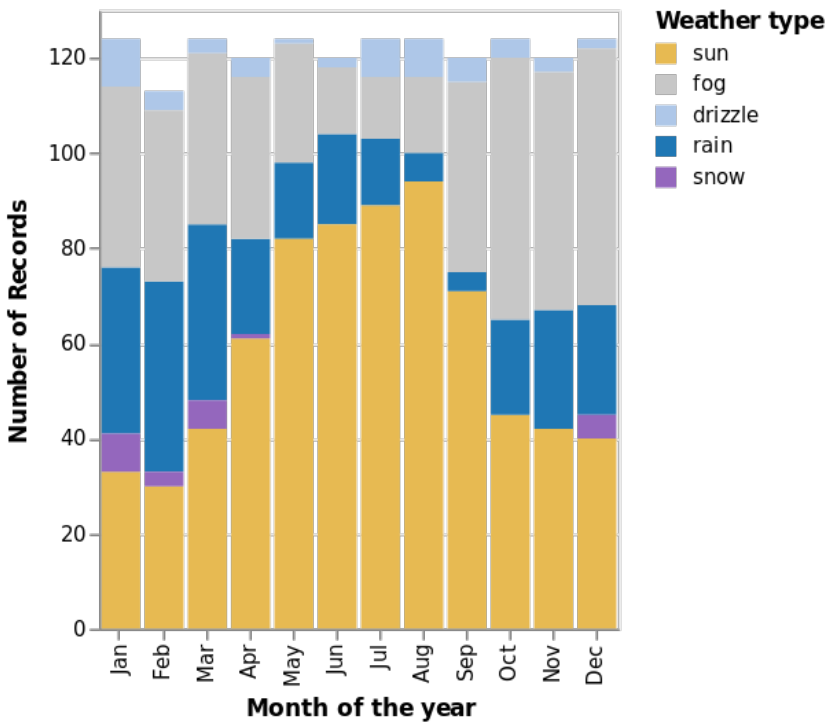
```
alt.Chart(df).mark_line().encode(
    alt.X("date:T", timeUnit="yearmonth"),
    alt.Y("max(temp_max)"),
)
```



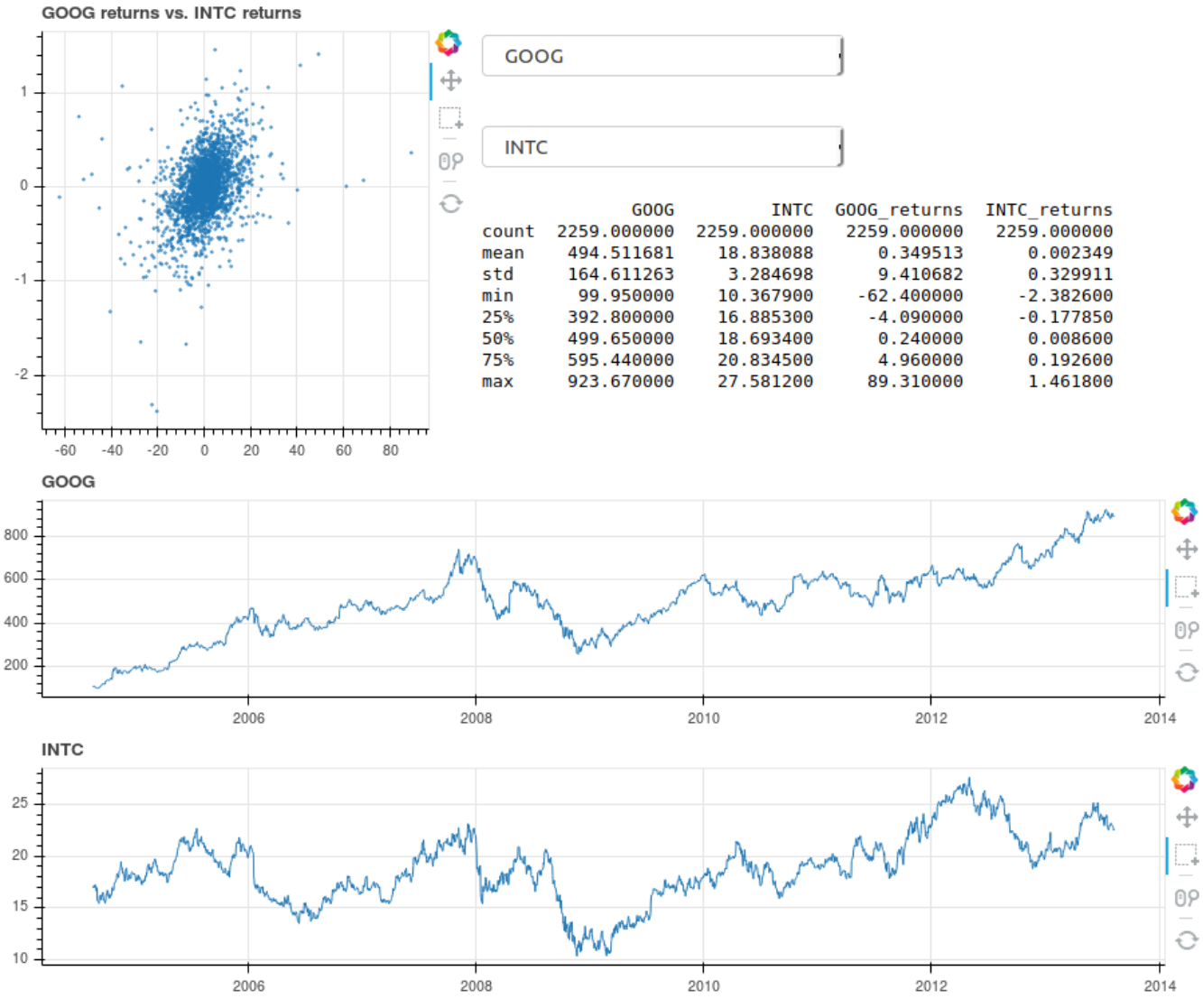
Tipo de día

Código

```
alt.Chart(df).mark_bar().encode(
    x=alt.X("date:N", timeUnit="month"),
    y="count()", color=alt.Color("weather",
    legend=alt.Legend(title="Weather type"), scale=scale),
)
```

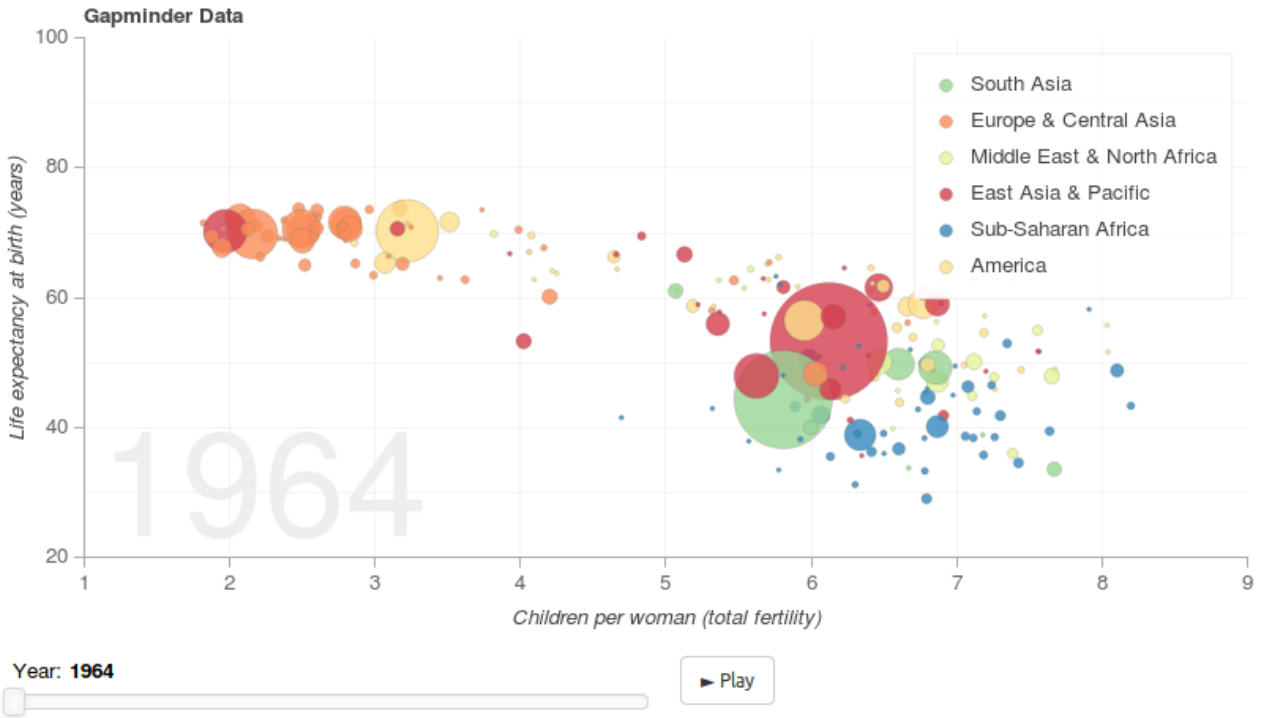


Demo de diagrama interactivo



	GOOG	INTC	GOOG_returns	INTC_returns
count	2259.000000	2259.000000	2259.000000	2259.000000
mean	494.511681	18.838088	0.349513	0.002349
std	164.611263	3.284698	9.410682	0.329911
min	99.950000	10.367900	-62.400000	-2.382600
25%	392.800000	16.885300	-4.090000	-0.177850
50%	499.650000	18.693400	0.240000	0.008600
75%	595.440000	20.834500	4.960000	0.192600
max	923.670000	27.581200	89.310000	1.461800

Demo de diagrama interactivo



Índice

1. Python para la ciencia
2. Python y la programación web

Python y la programación web

Django

- Entorno web Python más popular.
- Inspirado en Rails pero más explícito.
- Muy integrado.

Flask

- Entorno web más sencillo.
- Menos funcionalidad, extensible.
- Menos funcionalidad.

Programa en Flask

Ejemplo: hello world

```
from flask import Flask
app = Flask(__name__)

@app.route("/")
def hello():
    return "Hello World!"

if __name__ == "__main__":
    app.run()
```

Ejecutándolo

```
$ python ejemplos/web.py
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Cómo funciona

Asignar

1. @app.route permite asociar una URL una función.
2. La función devuelve el resultado como respuesta de la URL.

Parámetros

- La función puede recibir parámetros pasados por la URL.
- Notación: <nombre>.

Ejemplo de parámetro

```
@app.route("/saluda/<name>")
def hello(name):
    return "Hola␣{}".format(name)
```

Se puede combinar

Ejemplo

```
@app.route("/")
@app.route("/index")
def index():
  print("Pagina␣principal")
```

Otro ejemplo

```
@app.route("/saluda/<name>")
@app.route("/saluda")
def hello(name="desconocido"):
    return "Hola␣{}".format(name)
```

Templates

Plantilla

- La salida suele ser código HTML complejo.
- Se usa CSS personalizado (usando Bootstrap o similar).
- El diseñador crea la plantilla.
- La plantilla es HTML con referencias a variables (sintaxis especial).
- El programa Python carga la plantilla y sustituye los valores de variables.

Vamos a seguir con el ejemplo.

Flask con Template

templates/index.html

```
<html>
    <head>
        <title>{{ title }} - Prueba</title>
    </head>
    <body>
        <h1>Saludos, {{ user.username }}!</h1>
    </body>
</html>
```

Código

```
from flask import render_template
from app import app

@app.route("/")
def index():
    user = {"username": "Miguel"}
    return render_template("index.html", title="Home",
                           user=user)
```

Lógica en la plantilla

Plantilla: template/blog.html

```
<html>
    <head>
        {% if title %}
        <title>{{ title }}</title>
        {% else %}
        <title>Bienvenido</title>
        {% endif %}
    </head>
    <body>
        <h1>Saludos, {{ user.username }}!</h1>
        {% for post in posts %}
        <div><p>{{ post.author }}: <b>{{ post.body }}</b></p></div
        {% endfor %}
    </body>
</html>
```

Lógica en la plantilla

Controlador

```
from flask import render_template
from app import app

@app.route(/)
@app.route(/index)
def index():
    user = {username: Miguel}
    posts = [
        {
            author: {username: John},
            body: Beautiful day in Portland!
        },
        {
            author: {username: Susan},
            body: The Avengers movie was so cool!
        }
    ]
    return render_template(index.html, title=Home,
                           user=user, posts=posts)
```

Flask/Django ofrece mucho más

Ofrece

- Acceso a la Base de Datos.
- Validación de formularios.
- Soporte de peticiones REST.
- Componentes: Calendario, Google Maps, ...
- Y mucho más.

Ejemplo real

https://tflsgo.herokuapp.com/

Web Scraping

Web scraping

- Implica descargar datos de la web.

Vamos a ver un ejemplo real

`http://www.eweb.unex.es/eweb/maeb2015/?Conferencia___Sesiones_Especiales`

Programa

Main

```
def main(url):
    m = re.search("(.*\.es)/(.*)", url)
    prefix = m.group(1)
    html = requests.get(url)
    doc = lxml.html.fromstring(html.text)
    links = doc.xpath("//a/@href")
    ss_links = [link for link in links
                if "Sesiones_Especiales___S" in link]

    for pos, ss_link in enumerate(ss_links):
        extract_ss_info(append_prefix(prefix, ss_link), pos)


if __name__ == "__main__":
    urls = [
        "http://www.eweb.unex.es/eweb/maeb2015/?Conferencia___Sesi
    ]
    for url in urls:
        main(url)
```

Programa

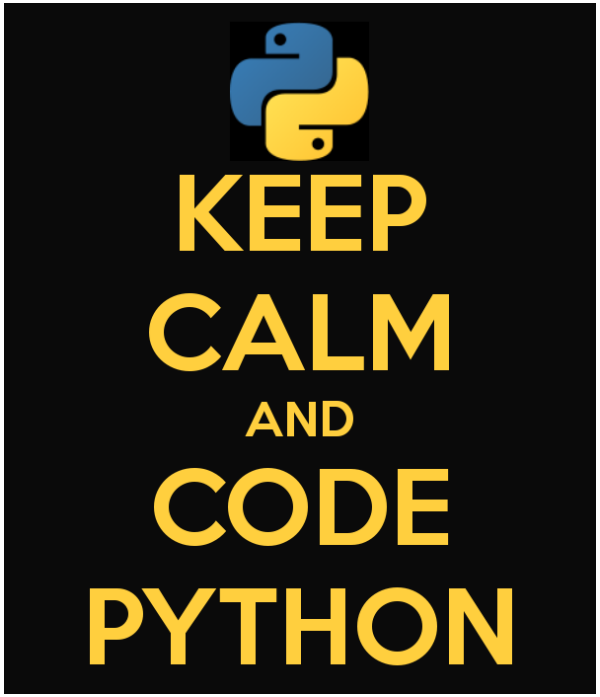
Funciones

```
def extract_ss_info(url, previous):
    ss = requests.get(url)
    ss.encoding = "utf-8"
    doc = lxml.html.fromstring(ss.text)
    titles = doc.xpath("//h3/span/text()")

    if titles:
        title = titles[0]
    else:
        title = doc.xpath("//h3/text()")[0]

    expected = "S{}.".format(previous + 1)
    titles = doc.xpath("//em[contains(text(),"Organizador")]")
    assert (len(titles) == 1)
    elem = titles[0].getparent()
    while elem.tag != "p":
        elem = elem.getparent()
    ul = elem.getnext()
    authors = [author.text_content() for author in ul.xpath("li")]
    info = [title]
```

Resumiendo



Conclusiones

- Python es muy útil.
- Sintaxis sencilla.
- Librerías muy potentes.
  - Menos código.
- Programación divertida.