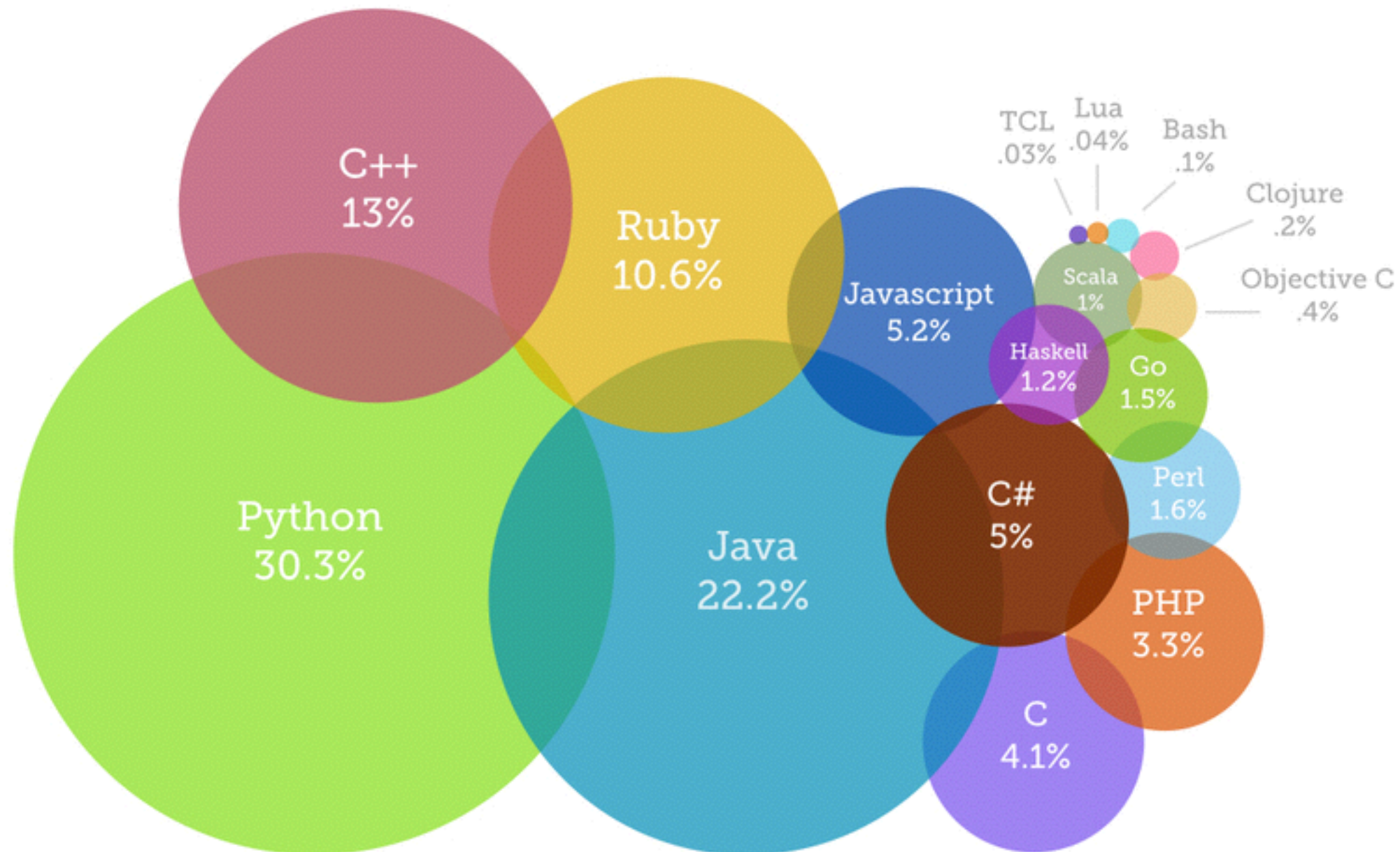




# Lenguajes de Programación

Alejandro Cárdenas-Avendaño

## Most Popular Coding Languages of 2014



# Lenguajes

## Compilados (Compiled)

- Deben ser compilados, es decir “traducidos” al código de máquina.
- Desarrollo más lento
  - Compilar -> Conectados -> Correr
- La sintaxis en general no es muy clara
- SON MÁS RÁPIDOS

**Fortran, C, C++, Java**

## Guión (Scripted)

- El código es corrido por un interpretador línea a línea.
- El archivo de texto es el programa
- Desarrollo más rápido de escribir y experimentar

**En línea**

**No compila Errores**

**Python, Javascript, Shell scripting.**

# C

- Hay que empezar todo desde cero, no trae nada.
- Se debe manejar la memoria
- Es muy portable
- No tiene Strings


# C++

- Está orientado a objetos
- La sintaxis no es muy clara
- No es actualizado
- Las librerías pueden ser llamadas a otros lenguajes

Tiempos críticos

# IDL

- ¡CUESTA!
- Fue impresionante en sus orígenes por las gráficas.
- Los intereses no están relacionados con ciencia como tal.
- Muchas librerías existen en IDL...

¿?  
  
print, a

# Python

- Es orientado a objetos
- Fácil de aprender, la sintaxis es “más natural”
- La memoria es manejada “internamente”
- El lenguaje ya tiene incluidas MUCHAS librerías y funcionalidades
- Está siendo continuamente actualizado
- Puede “citar” librerías escritas en C y C++

BATERIAS INCLUÍDAS

# C

```
#include <stdio.h>

int main(void) {
    printf("Hello, world!\n");
    return 0;
}
```

```
#include <stdio.h>
int main(void)
{
    double a, b, c, promedio;
    printf("Dame un número: ");
    scanf("%lf", &a); printf("Dame otro número: ");
    scanf("%lf", &b);
    printf("Y ahora, uno más: ");
    scanf("%lf", &c);
    promedio = (a + b + c) / 3;
    printf("El promedio es %f\n", promedio);
    return 0;
}
```

# Python

```
print 'Hello, world!'
```

```
a = float(raw_input('Dame un número:'))

b = float(raw_input('Dame otro número:'))
c = float(raw_input('Y ahora, uno más:'))
Promedio = (a + b + c) / 3

print 'El promedio es', promedio
```

# Editores de texto

- La mayoría del tiempo se está trabajando en el.
  - Fácil de Usar
  - Ayudas
    - Numeración de Líneas
    - Conozca nombres prohibidos
    - Señale la sintaxis
- Son como una religión

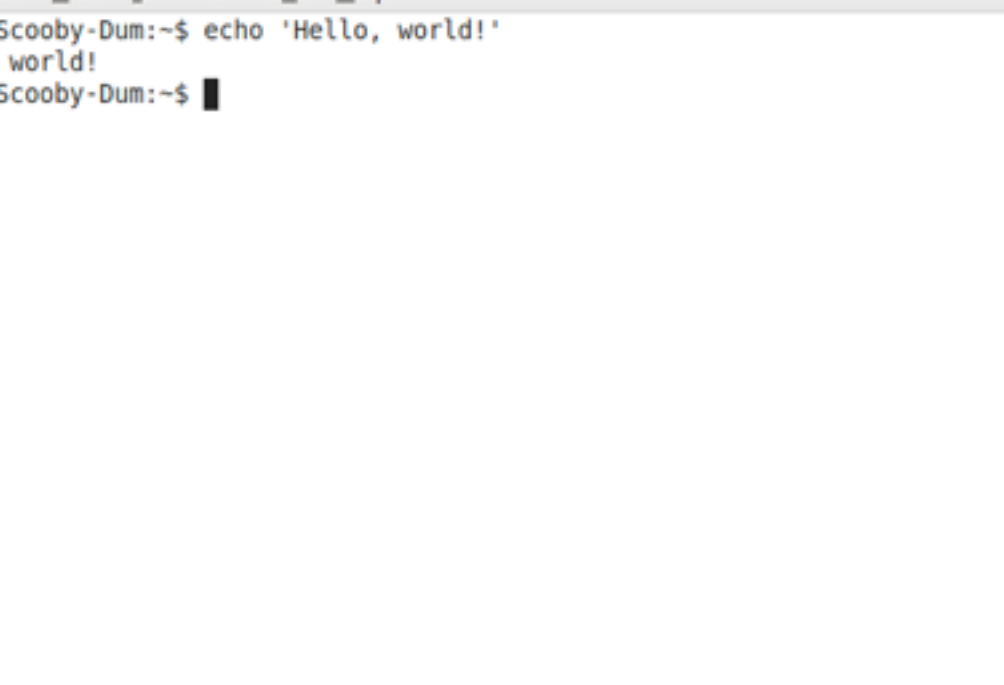


# Terminal

- Hay que personalizarlas

```
pilot:~/git/lufa-ftdi* ~/git/terminal/terminal.py --all /dev/ttyUSB*
/dev/ttyUSB0, 115200 baud
/dev/ttyUSB1, 115200 baud
/dev/ttyUSB2, 115200 baud
/dev/ttyUSB3, 115200 baud
/dev/ttyUSB4, 115200 baud
^C to exit
-----
[button pressed!]
[button pressed!]
[button pressed!]
[button pressed!]
[button pressed!]
[button pressed!]
[button pressed!]
[button pressed!]
[button pressed!]
[button pressed!]
[button pressed!]
[button pressed!]
[got byte X][[got byte X]ot byte X][got byte X][[got bgo[t byte X]te X]got byte X][got byte X]
```

- Que no se pixele
- Que sea fácil de leer
- Que no sea aburrida



```
bruce@Scooby-Dum: ~  
File Edit View Terminal Tabs Help  
bruce@Scooby-Dum:~$ echo 'Hello, world!'  
Hello, world!  
bruce@Scooby-Dum:~$
```

```
sergey@fremov: ~$ ssh root@127.0.0.1
Warning: Permanently added host 127.0.0.1 (SSH2-0.9.0).
root@kali:~# cat /etc/passwd | grep -v nologin | wc -l
191
root@kali:~# top
top - 12:51:17
Tasks: 191 total, 2 running
Load average: 0.46 0.68 0.91
Uptime: 6 days, 02:58:00
Mem: 12731/401840
Swap: 89/381280

PID USER      PR_ NI    VIRT   RES     SHR   S CPU% MEM%   TID% Command
15502 sergey  20  0 2076 48124 30344 S  9.0  1.2 2642.25 /usr/bin/X :0 -nr -verbose -auth /var/run/gdm/auth-for-gdm-pj7du
2570 sergey  9 -31 2236 37584 35344 S  1.0  0.9 1834.05 /usr/bin/pulseaudio --start --log-target=syslog
3324 sergey  20  0 51126 30396 11240 S  1.0  0.7 1801.08 am-applet -p /usr/share/avast-window-navigator/applets/taskmanag
3020 sergey  20  0 48620 13364 11200 S  0.0  0.3 5104.31 /usr/lib/gnome-panel/clock-applet --oaf-activate-tid=OAFIID:GNOM
2559 sergey  20  0 98016 14760 9080 S  1.0  0.3 4915.34 /usr/sbin/compsiz
5013 root      20  0 92016 15500 11024 S  0.0  0.9 1753.82 /usr/sbin/firestarter

root@kali:~# ps aux | grep -E "(sshd|rsync)"
sshd: root@pts/0 rlogin [root]
rsync: rsyncd [root]

root@kali:~#
```

# Computación científica

- **FLOPS**: floating point operations; Fortran, C
- **Estadística**: MAT\$LAB, R, Numpy
- **Big Data**: SP\$SS
- **Simbólico**: Map\$le, Mathem\$atica,
- **TODO**: Python, Ruby, Scala, Haskell

**Importante**

Tiempo

¿Cuál? -> Depende

**Minimizar**

Complejidad

Escribir

Modificar

# ¿Cómo debería ser?

- Llegar a un computador que nunca antes se haya visto
- Digitar “un par” de comandos
- Obtener el resultado deseado
- ...

¿Qué debo esperar de un leguaje?

# El proceso científico “Moderno”

- Aplicar a un grant
- Observar y Explorar fenómenos
- Generar un hipótesis
- Proponer modelos que expliquen el fenómeno
- Probar las predicciones del modelo
- Modificar la teoría y repetir todo el proceso
- Publicar

- Observar y Explorar fenómenos
- Permitir a cualquier persona manipular los datos fácilmente
- Muestre la información y permita hacer estadística básica
- Permita realizar variaciones en las visualizaciones
- Interactue con la comunidad

- Generar un hipótesis
- Aprendizaje no supervisado (Agrupamiento, redes neuronales)
- Reducción de dimensionalidad
- Compacto y que interactue con varios modelos

- Proponer modelos que expliquen el fenómeno
- Aprendizaje supervisado (Regresiones, clasificación)
- En genera una caja de herramientas
- Comunidad en desarrollo



- Probar las predicciones del modelo
- Pruebas automáticas (Significancias, Sensibilidad)
- Proponer pruebas
- Optimización
- Comparar

- Modificar la teoría y repetir todo el proceso
- ¡Notebook!
- Fácil de corregir (por mí y por la comunidad)
- Sesiones interactivas

- Publicar
- Gráficos “bonitos” y “agradables”
- Estadística apropiada
- “Verificables” (Que la matemática sea “uno a uno”)
- “Reproducible” (código simple)
- “Repetible” (Código abierto, datos)

# ¿De qué estamos hablando?

- “La elegancia no es una opción” R. O’Keefe
- Una sintaxis:
  - Explícita
  - Concisa
  - Consistente
- Abstracciones “terrenales”
- Corra en múltiples núcleos
- Documentación
- Ejemplos
- Tutoriales

# Referencias

- Muna, D & Price-Whelan, A. SciCoder Workshop.  
[scicoder.org](http://scicoder.org)
- Peter Norvig, “What to demand from a Scientific Computing Language”, Mathematical Sciences Research Institute, 2010.
- Eric Mjolsness, Dennis DeCoste, “Machine Learning for Science: State of the Art and Future Prospects”, **Science** 14 September 2001, Vol. 293 no. 5537 pp. 2051-2055.