Temario de la clase

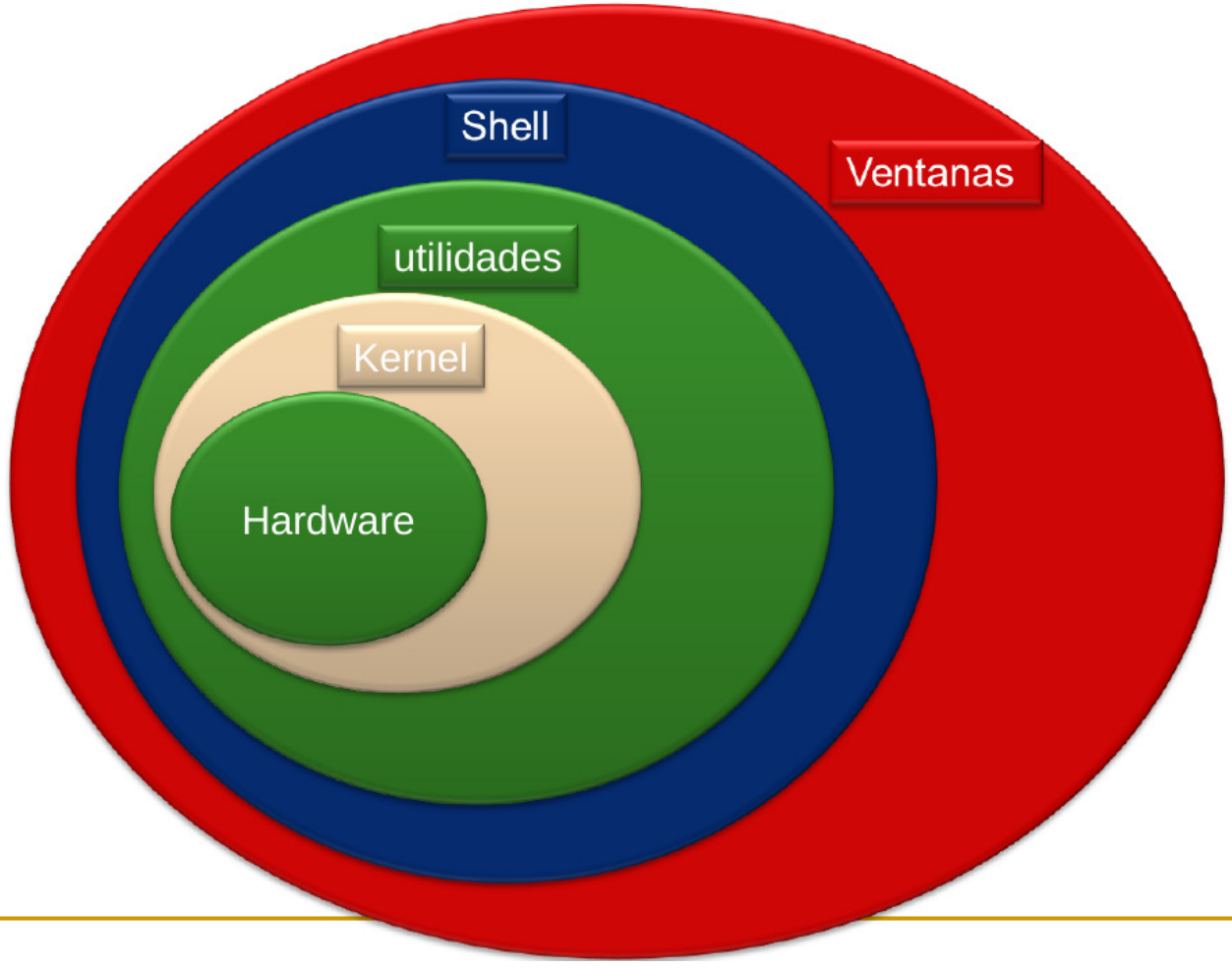
Temario

- Estructura de directorios y atributos de archivos en linux
- Comandos básicos de shell para la administración de archivos y directorios (ls, mkdir, mv, cp, etc.)
- Búsqueda de archivos y cadenas en archivos (find, grep)
- Scripts en bash
- Google colab. Introducción a su uso.

Esta funcionando la página web de la materia:
`https://pulidom.github.io/programacion`

En la solapa clases tienen la filminas nuevas.

Sistema Operativo



El sistema operativo propiamente es el kernel (nucleo), el resto incluyendo el administrador de archivos, bash, administrador de pantalla son aplicaciones.

Distribuciones del linux

- Debian
- Slackware
- Opensuse (HPC/Servidores).
- Ubuntu
- Lubuntu (Light-ubuntu).
- Centos, Fedora, Arch Linux, Mandriva, etc

Organización de los directorios y usuarios del linux

- user: Puede configurar sus propias aplicaciones y utilizar el sistema sin realizar cambios.
- root (superuser): Es el único que tiene permiso de cambiar variables del sistema operativo.

Estructura de directorios principales

- /root
- /home
- /usr
- /media
- /etc
- /proc

/home/[nombre-usuario] Es nuestro directorio de cabecera y alli tendremos nuestra estructura interna.

Permisos de los archivos

Permisos de los directorios o archivos: Lectura(r), Escritura(w), Ejecución(x)

El único lugar que tendremos permiso de escritura es el /home/[nombre-usuario]
En el resto de los directorios generalmente se tiene permiso de lectura pero no de escritura.

Hay archivos que pueden ejecutarse (con instrucciones que entienda la máquina). Todos los directorios son ejecutables.

Terminal: bash/c-shell

Es un lenguaje de programación que sirve para manejar a la computadora a través de cómandos básicos. Generalmente, se utiliza para configurar los sistemas, para manejar los dispositivos, manejo de usuarios, etc.

Lenguajes de programación: Interpretés y Compiladores.

Existen dos tipos de forma de dar las instrucciones en una terminal bash:

1. Interactiva: Ejecutan cada instrucción.
2. Script: Es un programa (conjunto de instrucciones) que se guarda en un archivo. El archivo/file debe tener permiso de “ejecución”.

ls: list

El comando ’ls’ produce una lista de los archivos del directorio donde nos encontramos.

```
$ ls [directorio]
```

nos lista los archivos dentro del directorio especificado.

```
$ ls [file]
```

nos lista el file especificado (si es que se encuentra alli o da un error si el file no existe).

Modificatorios/Opciones del comando:

```
$ ls -t
```

Lista los archivos ordenados por el último modificado primero.

Generalmente las opciones o modificatorios van con un ’-’

Atributos de un archivo

```
$ ls -l
```

Realiza la lista de archivos con los atributos de cada archivo.

Atributos de un archivo:

- Permisos (lectura, escritura, ejecución).
- Usuario y grupo (de usuarios) del archivo
- Tamaño de los archivos en bytes.
- Última fecha de modificación del file

El asterisco (comodín del poker)

```
$ ls a*
```

Lista todos los archivos que comienzan con ’a’

```
$ ls /usr/*.txt
```

Lista todos los archivos con extensión txt que se encuentran en el directorio usr

```
$ ls /usr/a*.txt
```

Igual que el anterior pero solo los que empiezan con a

```
$ ls /usr/a?.txt
```

El signo de interrogación es un comodín de un solo caracter.

cd: Change Directory

cd cambia el directorio.

Si queremos ir a un directorio específico:

```
$ cd /usr/local
```

Si queremos volver en un directorio de /usr/local a /usr:

```
$ cd ../
```

(los dos puntos vuelven para atrás en la estructura de directorio).

Si queremos ir a nuestro home directory /home/[usuario] usar la tilde:

```
$ cd
```

o

```
$ cd ~
```

Si queremos ir a algun directorio de nuestro home,

```
$ cd ~/programacion/guia1
```

(la tilde es una abreviatura de /home/usuario)

Mover, copiar, borrar archivos

Para mover un archivo o directorio: mv (move)

```
$ mv [file] [directory]
```

```
$ mv [directory] [directory]
```

Para copiar un archivo o directorio: cp (copy)

```
$ cp [file] [directory]
```

Para borrar archivos o directorio: rm (remove) NO se recomienda

```
$ rm -i [file]
```

La opción -i hace interactivo (pregunta si realmente lo quieren borrar).

Elimina totalmente los archivos o directorios (no se pueden recuperar).

Alternativa es: mover al trash [Directorio con archivos que se borran]

```
$ mkdir ~/trash
```

```
$ mv file ~/trash/
```

Varios

- man man=manual, nos da información sobre el comando, la instrucción e.g. man ls.
- mkdir Crea un directorio. MaKe DIRectory
- rmdir Borra un directorio. ReMove DIRectory
- pwd: muestra el nombre del directorio de trabajo (Print Working Directory)
- touch: Si el archivo no existe, lo crea. Si ya existía le cambia el tiempo al actual.
- echo: Imprime/Muestra en pantalla. echo 'Hola' echo Hola echo $HOLA
  Cualquier palabra con un $ se esta refiriendo a una variable del sistema $var (variable var). Entonces imprime el contenido de la variable 'HOLA' (si no esta definida no imprime nada).

Comando chmod

chmod: Change file mode. Cambia los permisos de un archivo.
Obviamente solo nos permite cambiar los permisos de los archivos que somos propietarios.
e.g. chmod u+w file, Primer carácter: a,u,g,o Segundo carácter: + o -
Tercero: r,w,x,X

chown: Change owner nos permite cambiar al propietario de un archivo
`$ chown usuario:users programacion` [Hace que programación pertenezca al 'usuario' y al grupo 'users'] (solo para archivos dentro de nuestro directorio)

Comando find

find Encontrar, es decir se utiliza para buscar archivos o directorios.

```
$ find /home/[usuario] -name "*.txt"
```

En este caso busca todos los archivos que tengan como extensión txt en nuestro home directory.

Comando grep

grep Busca en un archivo o grupo de archivos, una palabra o patrón.
Sintaxis: `$ grep -[opciones] [patron] [archivo]`

Supongamos queremos encontrar en un conjunto de archivos donde tenemos la información del experimento “Maxwell”.
Entonces lo que hacemos es buscar el patron: ’Maxwell’ en los directorios exp*.txt.

```
$ grep ’Maxwell’ exp*.txt
```

Este comando imprimirá como salida todas las líneas en los archivos exp*.txt que tengan la palabra Maxwell (El linux distingue entre mayúsculas y minúsculas)
Si se quiere buscar palabras con mayúsculas o minúsculas (ambas), maxwell, MAXWELL o Maxwel, agregar la opción -i.

Redireccionamiento de la salida de un comando

Existen comandos para redireccionar la salida de un comando.

'>' Manda la salida de un comando a un archivo:

Ejemplo:

```
$ ls > listado.txt
```

Guarda la salida del comando en un archivo llamado listado.txt. Si no esta lo genera al archivo. De lo contrario lo sobreescribe.

'|' manda la salida a otro comando Ejemplo:

```
$ ls *.txt | grep 'Ejercicio'
```

Busca en todos los archivos .txt la palabra Ejercicio. Es decir lo que hace el comando | es pasarle la lista de archivos encontrados al 'grep'.

Script

Para generar un script abrimos/creamos un archivo, prueba.sh (la extensión .sh nos indica por convención que esta escrito en bash/shell, pero se puede usar cualquier cosa)
Luego conviene poner en la primera línea del archivo:

```
#!/bin/bash
```

Esta instrucción dice que lo que viene esta en lenguaje de programación 'bash'.
Script para que la computadora salude y se presente: saludo.sh

```
#!/bin/bash
echo 'Hola. Soy Alexa'
```

Ejecución del script

Por otro lado se deben asegurar de que el archivo tenga permiso de ejecución.

```
$ chmod a+x saluda.sh
```

Luego para ejecutarlo solo tienen que hacer:

`$ ./saluda.sh` (si estan en el mismo directorio)

Si estan en cualquier lugar:

```
$ ~/programacion/saluda.sh
```

(Se pone todo el path o camino)

Scripts con variables de entrada

Modificamos el programa anterior para que interactue con el usuario.

```
#!/bin/bash
echo 'Hola '$1'. Yo soy Alexa'
```

Ejecutando como: ./saluda.sh [tuNombre]

Es decir, que $1 es una variable de entrada en el script.

Cálculos con el bash

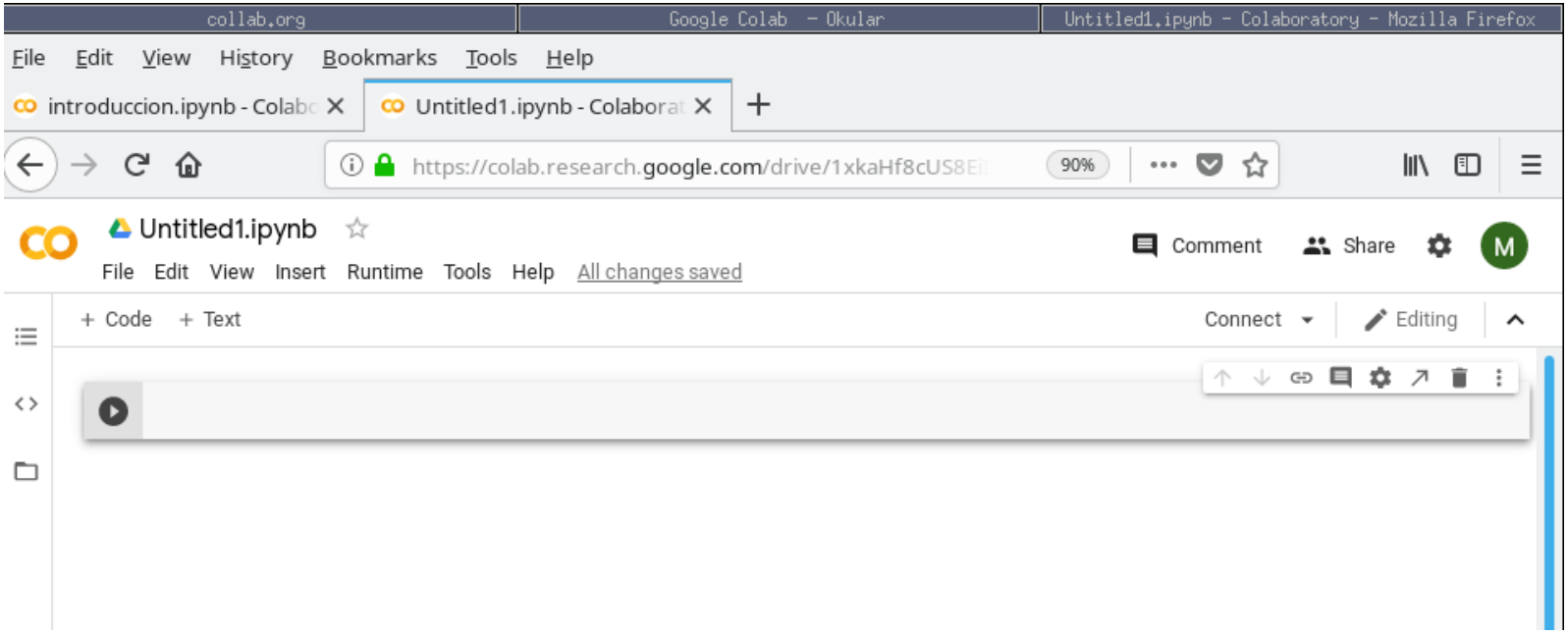
Los cálculos en bash NO son recomendados. Vamos a utilizar python que es mucho mas potente. Bash tiene una forma bastante primitiva de cálculos:

```
$ myvar = 2
$ expr $myvar + 1
$ expr $myvar / 3
$ expr $myvar \* 3
```

Google colab

Herramienta de google que pone a disposición una máquina virtual en linux para que nosotros podamos utilizar desde un navegador:

- En un navegador ir al sitio https://colab.research.google.com
- Deben tener abierta la cuenta de gmail solicitada para que el google les permita trabajar.
- Generen una nueva notebook.



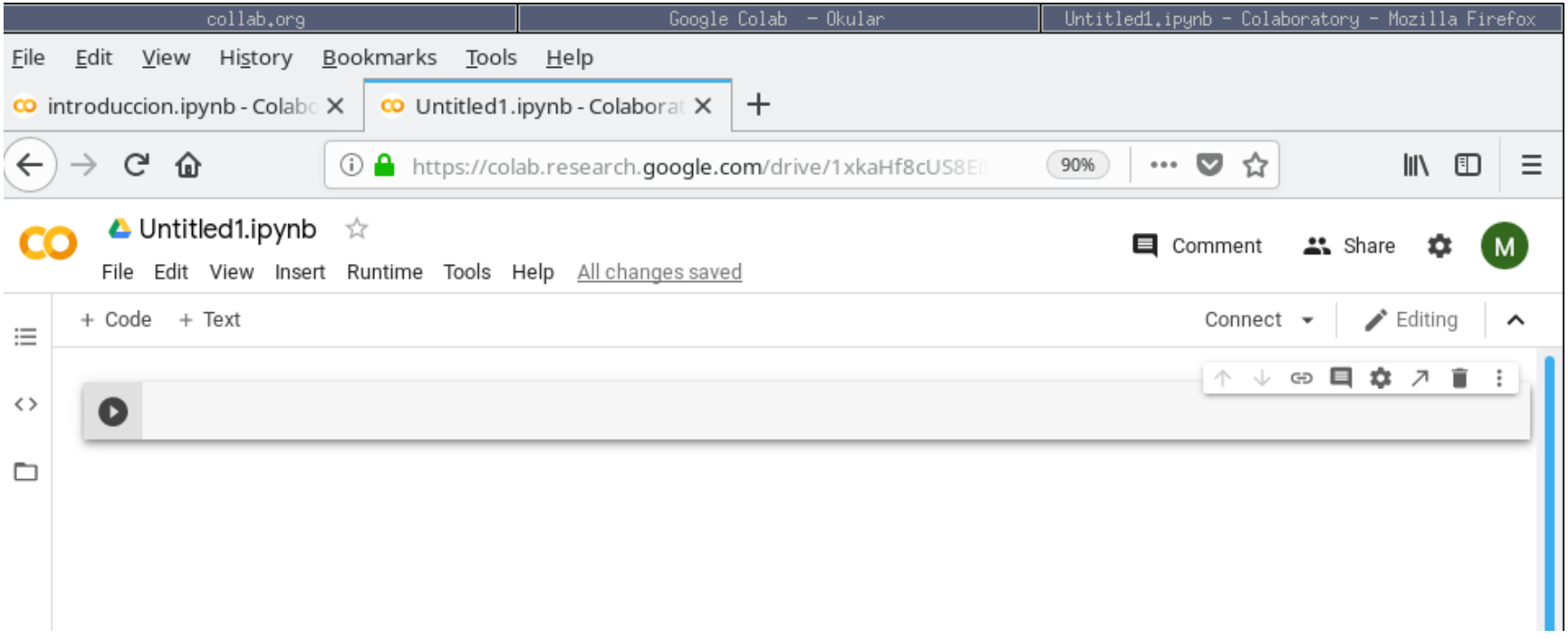
Características de google colab

Google colab esta basado en una aplicación de python: jupyter notebook

- El formato de escritura sigue el estandard de jupyter.
- Entiende y trabaja en el lenguaje python.
- Permite escribir programas en forma amigable. (literate programming)

En una notebook (cuaderno de laboratorio) se mezclan códigos, con textos y también las salidas/resultados de los códigos tanto textos como gráficos.

Características de la template



El primer borrador le pone Untitled1.ipynb. Renombren la notebook y le ponen introduccion.ipynb

La extensión es obligatoria y es la definida para las notebooks de jupyter.

Primer programa muy tonto en colab

Introduzcan en una celda las siguientes instrucciones:

```
a=15.
b=38.
print(a,'x',b,'es: ',a*b)
```

Para ejecutar la celda:

- Clickear la regla de la derecha de la celda
- Menu Runtime. Run focused Cell.
- Ctrl + Enter (Mas rápida)

La celda entonces es un conjunto de instrucciones parte de la notebook.

Generar una nueva celda

Al final de la solapa de la celda que ya usaron tiene una opción de introducir código o texto (o debajo del menú). Agregar una celda nueva y tipear:

```
c=b/a
```

```
print(c)
```

Ejecutar (regla de la derecha de la celda)

Realizó el cálculo con las variables que tenía en la celda anterior!.

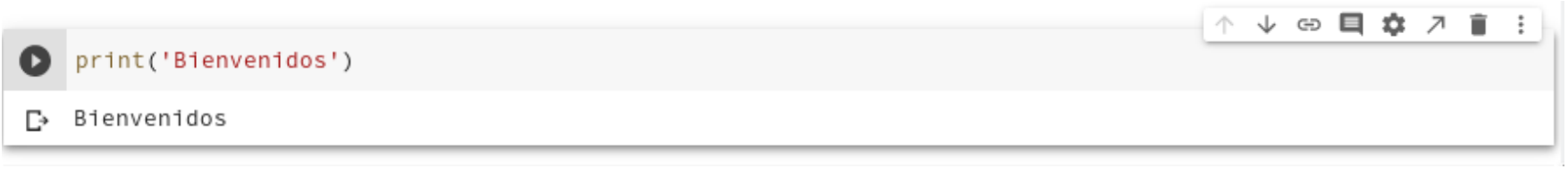
Es decir que cada una de las instrucciones que hemos ido introduciendo colab las deja en memoria.

Cuando creamos esta nueva celda las variables b y a, y todas las variables de las celdas anteriores que usemos ya quedaron definidas.

Navegación entre celdas

- Clickeando con el ratón en cualquiera de las celdas anteriores nos permite retomarlas.
- Cambien la variable a=5 en la primera celda y luego ejecutan la primera y la segunda celda.

También se puede cambiar el orden de las celdas, borrarlas, etc. Utilizando el menú que hay en el margen superior izquierdo de cada celda van a ver cada una de las funciones.



Guardado de archivos

Se pueden guardar (save) tanto en el Google Drive como también en github (sitio web para repositorios de software).

Vamos a trabajar con Google Drive ya que el github requiere de conocimientos mas complejos de versionador de software (git).

- Ir al Menu, Solapa de File y luego van a ver la opción "Save a copy in drive"
- Guardar el archivo como introduccion.ipynb
- Abrir el google drive y ver que lo ha guardado en el directorio Colab Notebooks

Linux y la shell en colab

Por detrás de google-colab hay una máquina virtual en linux (ubuntu) con todas las funciones de linux incluyendo una terminal.
Para ejecutar instrucciones de bash/terminal en colab deben introducir el signo de admiración !
Ejemplos: En las celdas hagan

```
!ls
!cd /; ls
```

Es decir que todos los comandos de bash los tienen en colab.

```
! ls /bin
```

les va a a dar una lista de los comandos (ejecutables binarios) que pueden utilizar. Como verán todos los comandos bash estan allí, también hay comandos para manejar el sistema (e.g. journalctl, systemctl, systemd-*, mount, lsmod, etc.)

Montado del google drive en la máquina virtual de colab

Como es una máquina virtual todo lo que esta allí (la estructura de directorios/archivos etc) desaparece una vez que dejan de usar la aplicación.
Para que puedan guardar cosas pueden montar el google drive de uds. en la máquina virtual:

```
from google.colab import drive
drive.mount('/content/gdrive')
```

Les va a pedir una clave de autentificación ya que google colab va a acceder a los archivos de google drive (otra aplicación).
Alternativa: Vayan a Tools pongan m (mount drive) y allí les introducirá el comando.

Para escribir todo una celda de bash

Existe un conjunto de lenguajes que se pueden utilizar en una celda en colab a través de magics.

Los lenguajes que entiende son varios: html, latex, javascript, ruby, shell, bash.

```
%%bash
```

Todo lo que sigue en la celda será en lenguaje bash.

```
%%bash
  echo 'Entro a la shell'
  cd /bin
  echo 'Cambio de directorio ahora estoy en:'
  pwd
  echo 'Archivos que hay en este directorio:'
  ls
  echo
  echo 'Salgo de la shell'
```