Introducción a Python

02MBIF – Programación con Python y R

Universidad Internacional de Valencia

De:
Planeta Formación y Universidades

Introducción a la programación

—¿Qué sabes de este asunto?— preguntó el Rey a Alicia.
—Nada— dijo Alicia.
—¿Absolutamente nada?— insistió el Rey.
—Absolutamente nada— dijo Alicia.
—Esto es importante— dijo el Rey, volviéndose hacia los jurados.

Alicia en el país de las maravillas, Lewis Carroll

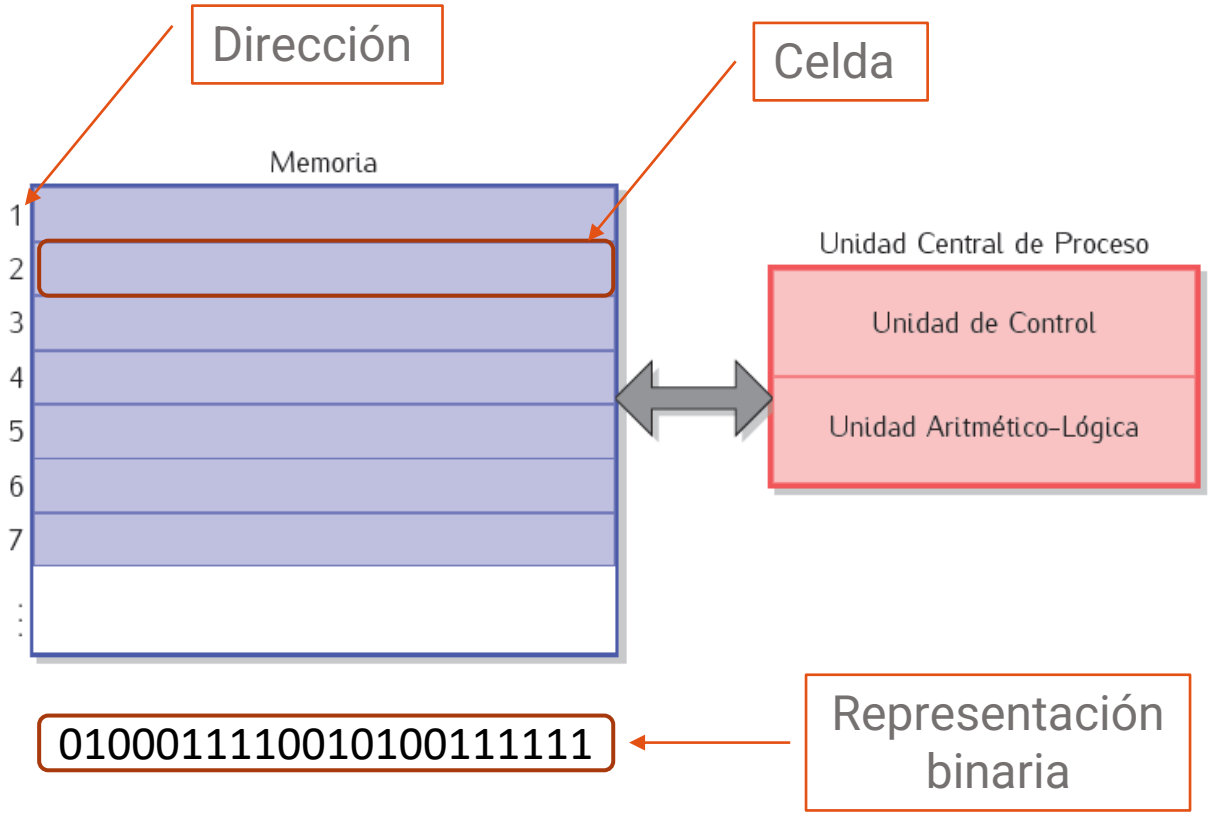
El objetivo de este curso es enseñarte a **programar**, esto es, a **diseñar algoritmos** y expresarlos como **programas** escritos en un **lenguaje de programación** para poder ejecutarlos en un **computador.**

Introducción a la programación: computadores

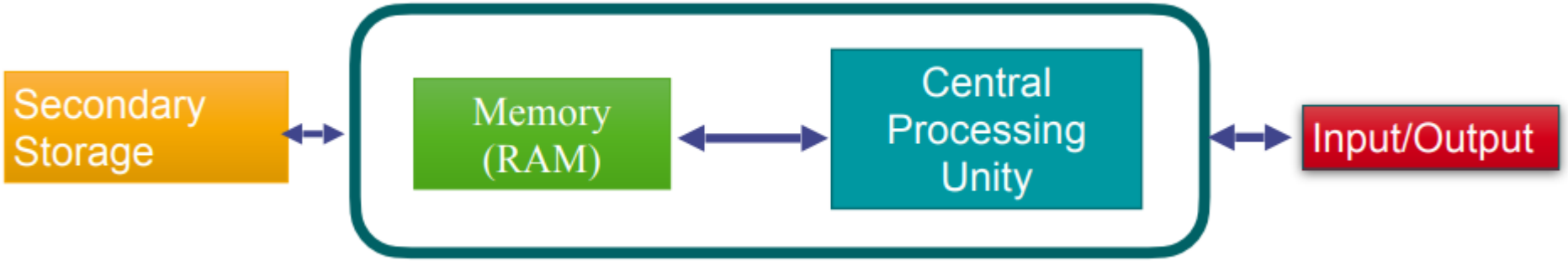
El diccionario de la Real Academia define computador electrónico como «*Máquina electrónica, analógica o digital, dotada de una* ***memoria*** *de gran capacidad y de métodos de tratamiento de la información,* ***capaz de resolver problemas matemáticos y lógicos*** *mediante la utilización automática de programas informáticos*»

Dos elementos básicos:

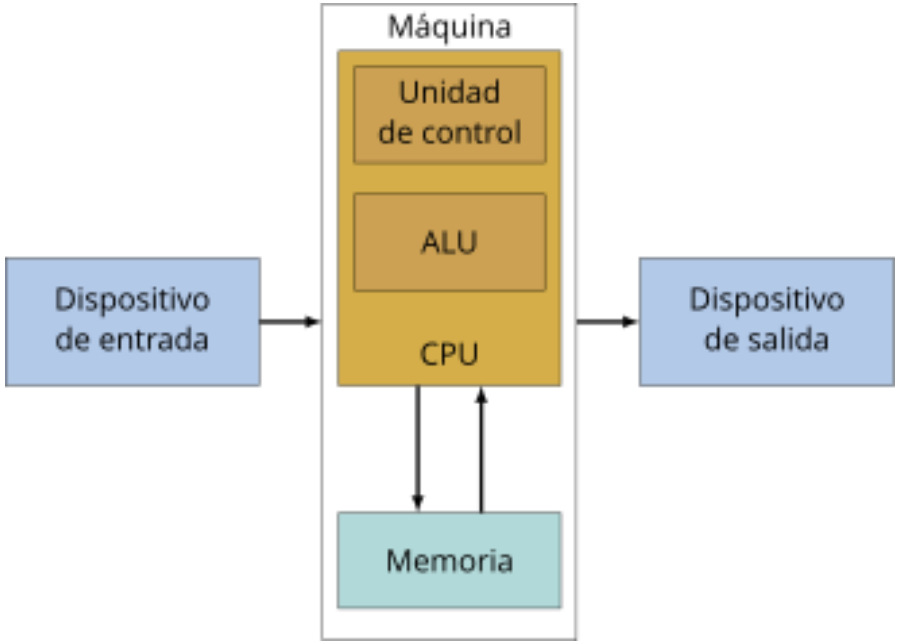
- La **memoria**. Permite almacenar cualquier tipo de datos: valores numéricos, textos, imágenes, etc.
- Dispositivo capaz de efectuar cálculos matemáticos y lógicos: la **Unidad Central de Proceso** (CPU).



Introducción a la programación: arquitectura de computadores



- **Unidad Central de Proceso (CPU)**
  - Ejecuta operaciones matemáticas.
- **Memoria (RAM)**
  - Almacena datos (limitados) para la CPU (4-32 Gigabytes)
  - Acceso rápido, pero no permanente.
- **Almacenamiento permanente (Storage)**
  - Acceso lento/gran capacidad (1000 Gigabytes).
  - Almacenamiento permanente de ficheros.
- **Entrada/Salida (Input/Output)**
  - Monitor/Teclado/Ratón/Tarjeta de Red



Introducción a la programación: codificación de la información

- **Codificación:**
  - Asocia signos con los elementos de un conjunto.
  - Código binario: {0, 1}
  - Bit: variable que sólo puede tomar dos valores binarios.
  - Byte: secuencia de 8 bits (00010101)
    - Permite codificar 256 significados.
- **El código binario es un sistema de numeración posicional que permite representar cualquier número.**
- **El código binario permite representar texto**
  - Tabla ASCII (H<->01001000)

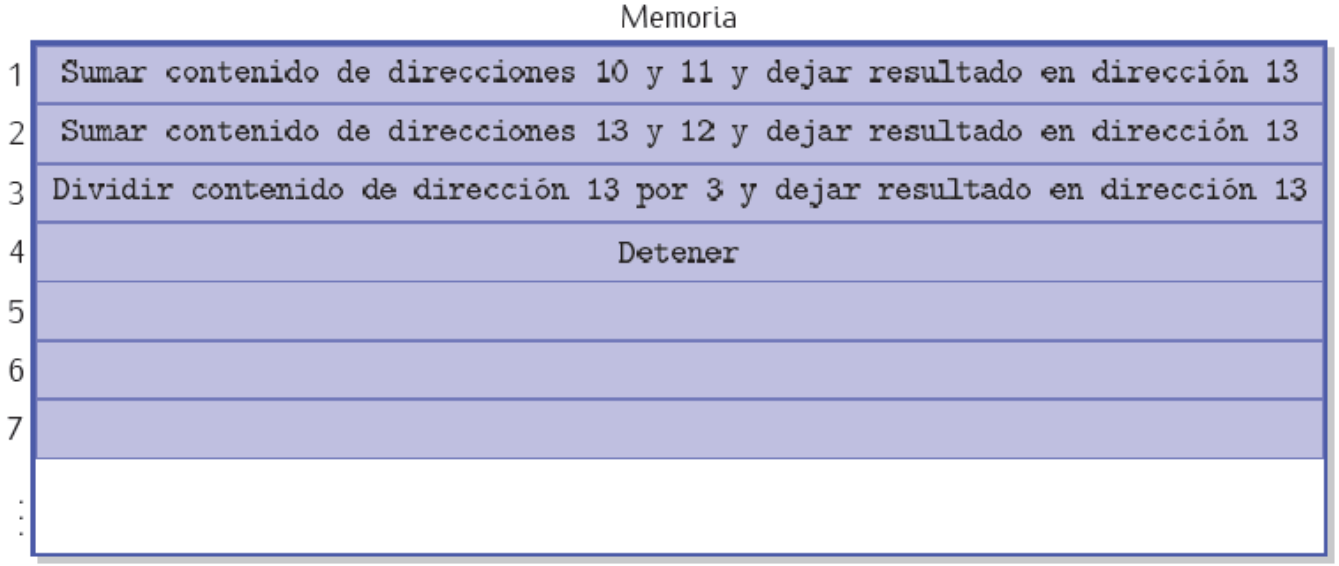
Caracteres ASCII de control		
00	NULL	(carácter nulo)
01	SOH	(inicio encabezado)
02	STX	(inicio texto)
03	ETX	(fin de texto)
04	EOT	(fin transmisión)
05	ENQ	(consulta)
06	ACK	(reconocimiento)
07	BEL	(timbre)
08	BS	(retroceso)
09	HT	(tab horizontal)
10	LF	(nueva línea)
11	VT	(tab vertical)
12	FF	(nueva página)
13	CR	(retorno de carro)
14	SO	(desplaza afuera)
15	SI	(desplaza adentro)
16	DLE	(esc.vínculo datos)
17	DC1	(control disp. 1)
18	DC2	(control disp. 2)
19	DC3	(control disp. 3)
20	DC4	(control disp. 4)
21	NAK	(conf. negativa)
22	SYN	(inactividad sínc)
23	ETB	(fin bloque trans)
24	CAN	(cancelar)
25	EM	(fin del medio)
26	SUB	(sustitución)
27	ESC	(escape)
28	FS	(sep. archivos)
29	GS	(sep. grupos)
30	RS	(sep. registros)
31	US	(sep. unidades)
127	DEL	(suprimir)

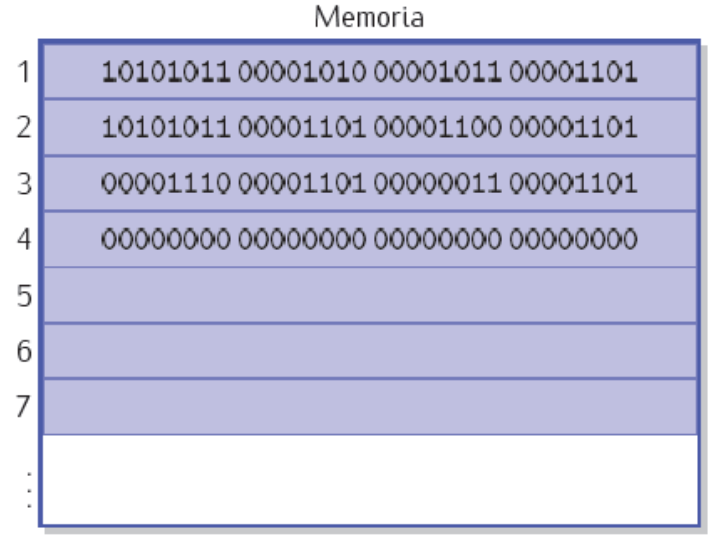
Caracteres ASCII imprimibles					
32	espacio	64	@	96	`
33	!	65	A	97	a
34	"	66	B	98	b
35	#	67	C	99	c
36	$	68	D	100	d
37	%	69	E	101	e
38	&	70	F	102	f
39	'	71	G	103	g
40	(	72	H	104	h
41	)	73	I	105	i
42	*	74	J	106	j
43	+	75	K	107	k
44	,	76	L	108	l
45	-	77	M	109	m
46	.	78	N	110	n
47	/	79	O	111	o
48	0	80	P	112	p
49	1	81	Q	113	q
50	2	82	R	114	r
51	3	83	S	115	s
52	4	84	T	116	t
53	5	85	U	117	u
54	6	86	V	118	v
55	7	87	W	119	w
56	8	88	X	120	x
57	9	89	Y	121	y
58	:	90	Z	122	z
59	;	91	[	123	{
60	<	92	\	124	|
61	=	93	]	125	}
62	>	94	^	126	~
63	?	95	_		

ASCII extendido (Página de código 437)							
128	Ç	160	á	192	└	224	Ó
129	ü	161	í	193	┴	225	ß
130	é	162	ó	194	┬	226	Ô
131	â	163	ú	195	├	227	Ò
132	ä	164	ñ	196	─	228	õ
133	à	165	Ñ	197	┼	229	Õ
134	å	166	ª	198	ã	230	µ
135	ç	167	º	199	Ã	231	þ
136	ê	168	¿	200	╚	232	Þ
137	ë	169	®	201	╔	233	Ú
138	è	170	¬	202	╩	234	Û
139	ï	171	½	203	╦	235	Ù
140	î	172	¼	204	╠	236	ý
141	ì	173	¡	205	═	237	Ý
142	Ä	174	«	206	╬	238	¯
143	Å	175	»	207	¤	239	´
144	É	176	░	208	ð	240	≡
145	æ	177	▒	209	Ð	241	±
146	Æ	178	▓	210	Ê	242	‗
147	ô	179	│	211	Ë	243	¾
148	ö	180	┤	212	È	244	¶
149	ò	181	Á	213	ı	245	§
150	û	182	Â	214	Í	246	÷
151	ù	183	À	215	Î	247	¸
152	ÿ	184	©	216	Ï	248	°
153	Ö	185	╣	217	┘	249	¨
154	Ü	186	║	218	┌	250	·
155	ø	187	╗	219	█	251	¹
156	£	188	╝	220	▄	252	³
157	Ø	189	¢	221	¦	253	²
158	×	190	¥	222	Ì	254	■
159	ƒ	191	┐	223	▀	255	nbsp

Introducción a la programación: programas y lenguajes de programación

- La CPU es capaz de ejecutar acciones específicas mediante secuencia de **instrucciones** (acción simple).
- **Programa**: secuencia de instrucciones para ejecutar cálculos complejos.
- **Código máquina**: secuencias de instrucciones codificadas en binario. Depende de la CPU.
- **Ejemplo**: calcular la media de tres números almacenados en las posiciones de memoria 10, 11 y 12, y dejar el resultado en la posición 13.





Introducción a la programación: programas y lenguajes de programación

- **Lenguaje ensamblador:** utiliza un nemotécnico para representar cada instrucción en código máquina.

- ¿Cómo ejecutar este tipo de programa? Con la ayuda del **ensamblador. Es un programa traductor que lee los códigos nemotécnicos y los traduce instrucciones en código máquina.**

- **Lenguaje ensamblador**: conjunto de códigos nemotécnicos y las instrucciones que permiten combinarlos y representar direcciones y valores numéricos.
  - **Cada CPU tiene un lenguaje ensamblador propio.**
  - **Programar en ensamblador es complejo.**

```
SUM #10, #11, #13
SUM #13, #12, #13
DIV #13, 3, #13
FIN
```

```
    .data
msg:
    .string "Hello, World!\n"
len:
    .long . - msg
.text
.globl _start
_start:
    push $len
    push $msg
    push $1
    movl $0x4, %eax
    call _syscall
    addl $12, %esp
    push $0
    movl $0x1, %eax
    call _syscall
_syscall:
    int  $0x80
    ret
```

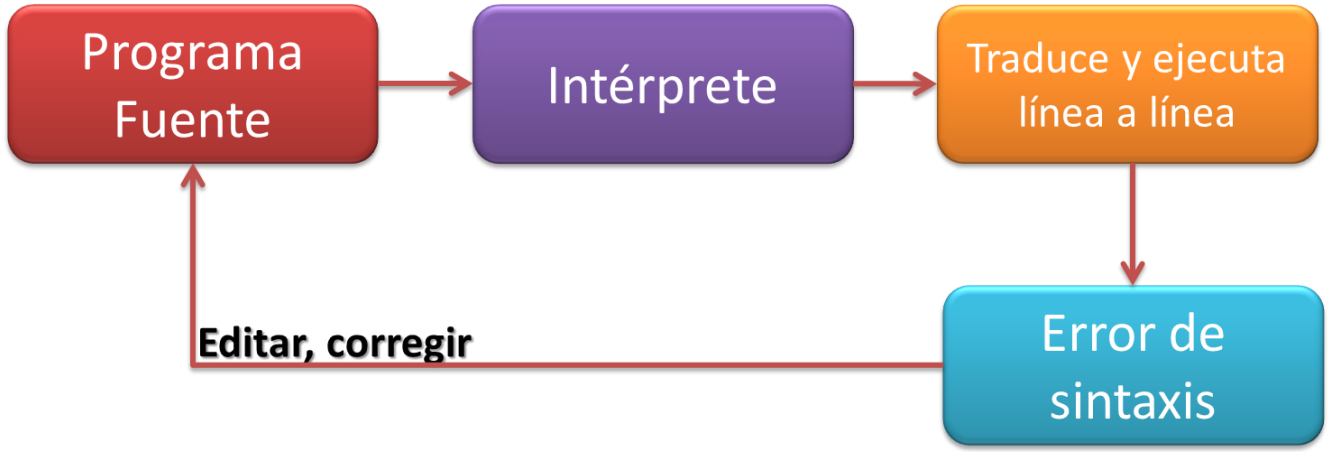
```
start:
  move.l #msg,-(a7)
  move.w #9,-(a7)
  trap #1
  addq.l #6,a7
  move.w #1,-(a7)
  trap #1
  addq.l  #2,a7
  clr -(a7)
  trap #1
  msg: dc.b "Hello, World!",10,13,0
```

Introducción a la programación: programas y lenguajes de programación

- **Lenguaje de programación de alto nivel:** lenguajes próximos al natural e independientes del ordenador concreto (Python).

- **Compiladores**: **lee completamente** un programa en un lenguaje de alto nivel y **lo traduce en su integridad** a un programa de código de máquina equivalente. Una vez traducido se puede ejecutar el programa de forma indefinida.

- **Intérpretes: lee** un programa escrito en un lenguaje de alto nivel **instrucción a instrucción** y, para cada una de ellas, efectúa una **traducción** a las instrucciones de código de máquina equivalentes y las **ejecuta inmediatamente.**



Python

Python es un **lenguaje de alto nivel de programación** **interpretado** cuya filosofía hace hincapié en la **legibilidad de su código**, se utiliza para **desarrollar aplicaciones de todo tipo**, ejemplos: Instagram, Netflix, Spotify, Panda3D, entre otros.

Se trata de un **lenguaje de programación multiparadigma**, ya que soporta parcialmente la **orientación a objetos**, **programación imperativa** y, en menor medida, **programación funcional**. Es un **lenguaje interpretado**, **dinámico** y **multiplataforma**.

Fuente: Wikipedia

¿Por qué Python?

- Python es **un lenguaje muy expresivo**, es decir, los programas Python son muy compactos: un programa Python suele ser bastante más corto que su equivalente en lenguajes como C.
- Python es **muy legible**.
- Python ofrece un **entorno interactivo** que facilita la realización de pruebas.
- El **entorno de ejecución** de Python detectar muchos errores que escapan al control de los compiladores.
- Puede utilizarse como **lenguaje imperativo o como lenguaje orientado a objetos**.
- Posee **un rico juego de instrucciones de datos**.

Python: historia

«Hace seis años, en diciembre de 1989, estaba buscando un proyecto de programación como hobby que me mantuviera ocupado durante las semanas de Navidad. Mi oficina estaría cerrada y no tendría más que mi ordenador de casa a mano. Decidí escribir un intérprete para el nuevo lenguaje de scripting que había estado ideando recientemente: un descendiente de ABC que gustaría a los hackers de Unix/c. Elegí el nombre de Python para el proyecto, encontrándome en un estado de ánimo ligeramente irreverente (y siendo un gran fan de Monty Python's Flying Circus)».

- Nació de la mano de **Guido van Rossum** mientras trabajaba para el Centro de las Matemáticas y la Informática de los Países Bajos.
- La **primera versión se publicó en 1991** en USENET.
- En la actualidad la **Python Software Foundation** es la responsable de velar por la evolución futura del lenguaje. (**www.python.org**).
- Un momento clave en la historia del lenguaje fue el 3/12/2008 fecha en la que se liberó la versión 3.0.
- Hace especial énfasis en su **capacidad de ser leído por otros programadores**, eso ha potenciado su uso en entorno científicos y de la ciencia de datos.

Python: lenguaje más popular

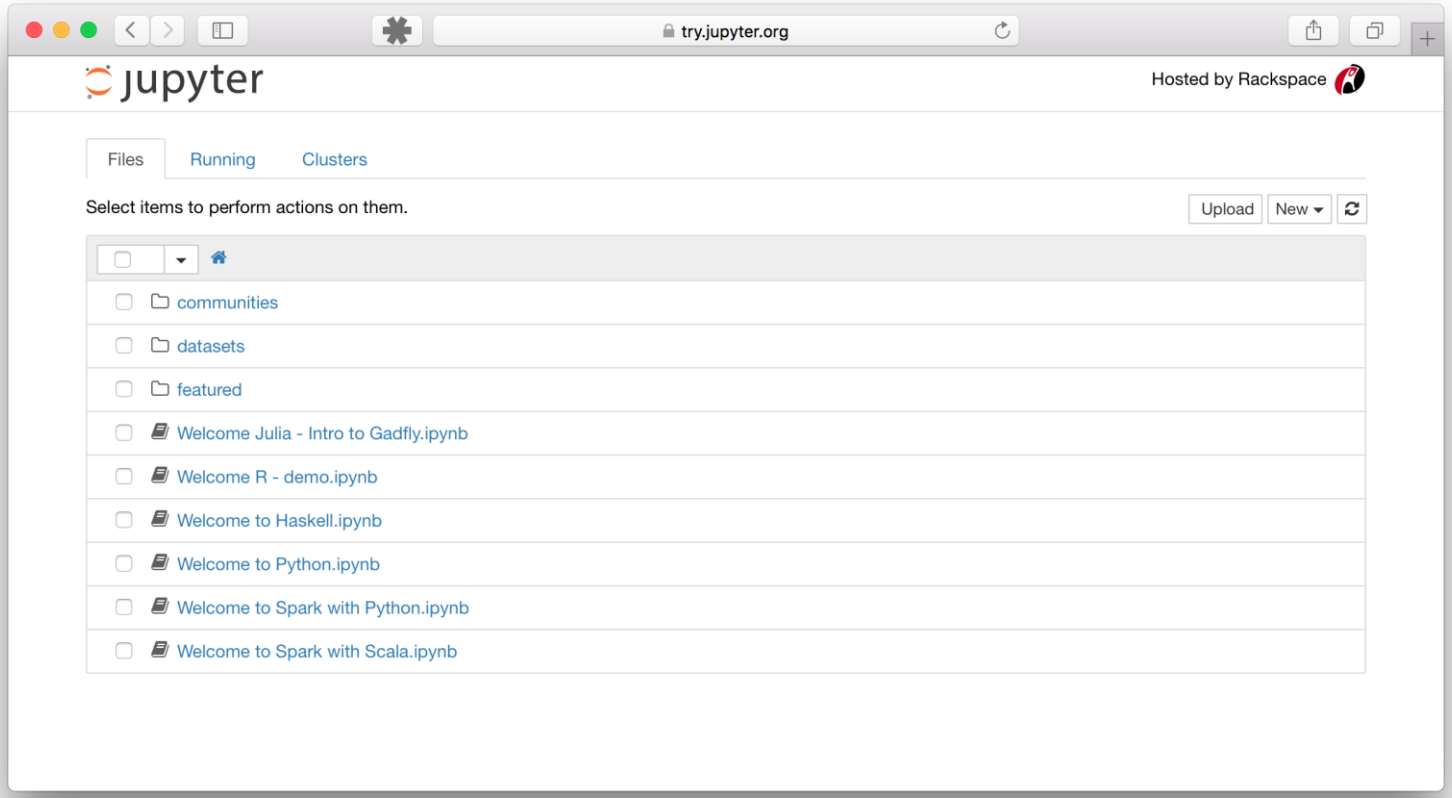
Fuente: https://spectrum.ieee.org/top-programming-languages-2022

Jupyter Notebook

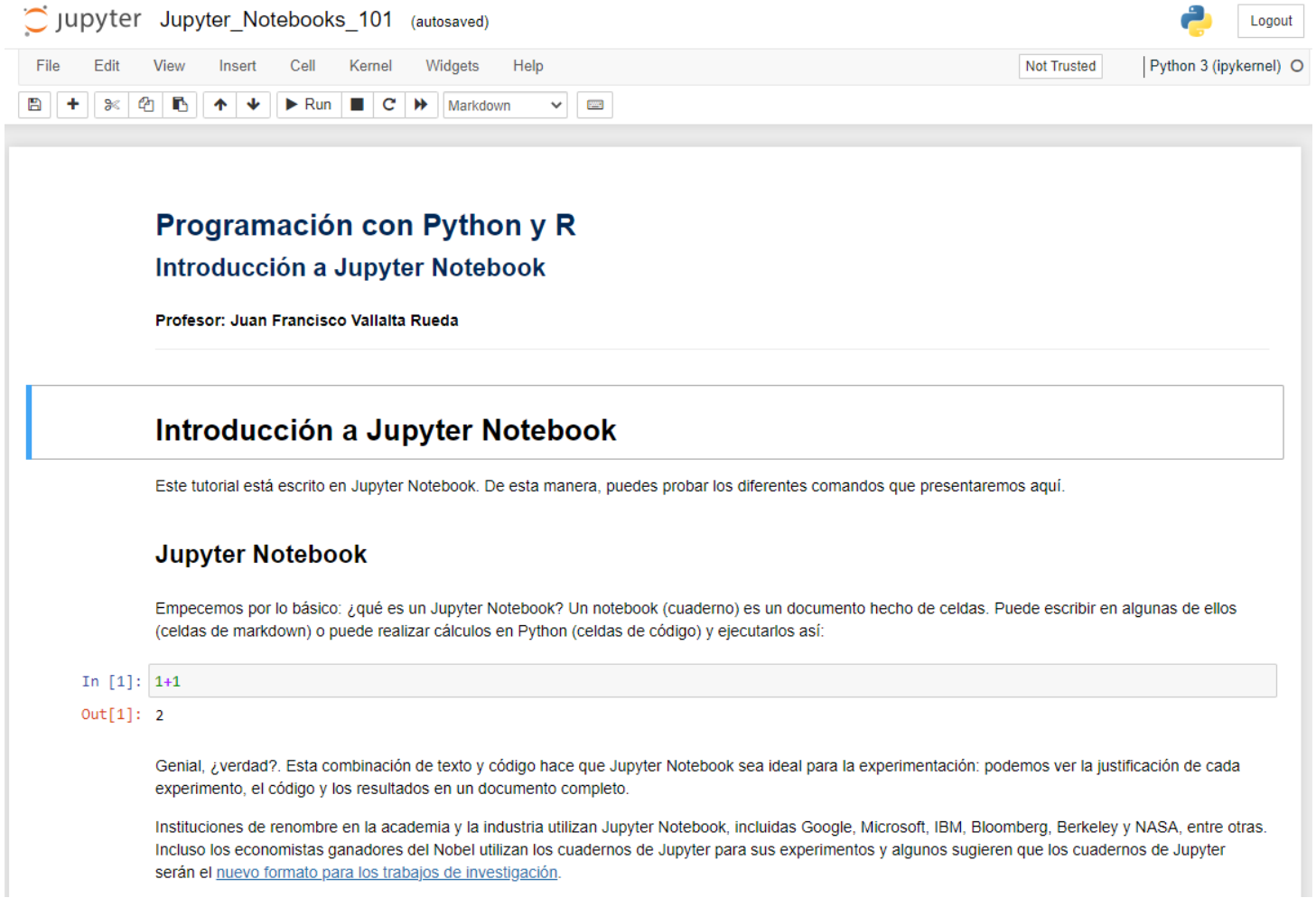
- Un **cuaderno** es un documento que se puede compartir que **combina código de computadora, descripciones en lenguaje sencillo, datos, visualizaciones enriquecidas como modelos 3D, cuadros, gráficos y figuras, y controles interactivos**.
- Un cuaderno, junto con un editor (Jupyter Notebook), **proporciona un entorno interactivo rápido** para crear prototipos y explicar el código, explorar y visualizar datos y compartir ideas con otros.

Jupyter Notebook: gestor de cuadernos

Un interfaz orientado a documentos que le permite crear, ver y ejecutar código en un Jupyter Notebook.



Jupyter Notebook: editor de cuadernos



Jupyter Notebook: primeros pasos

Jupyter Notebook

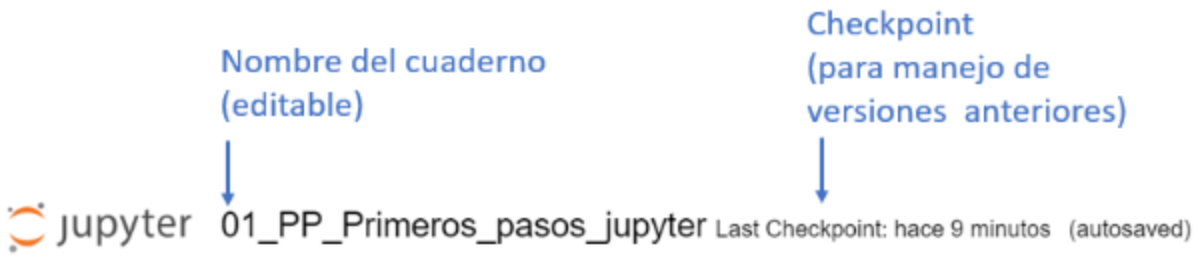
- Entorno de trabajo interactivo para programación con python.
- Permite ejecutar de manera selectiva pasos del código
- Permite insertar bloques de texto para documentar los pasos de ejecución

Ejecutar jupyter

Desde la terminal ejecutamos el comando en el directorio de trabajo que tiene los cuadernos:

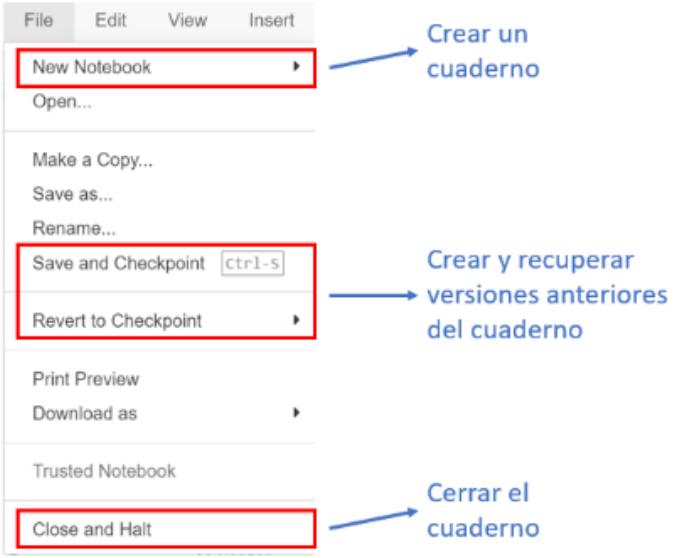
- **jupyter notebook**

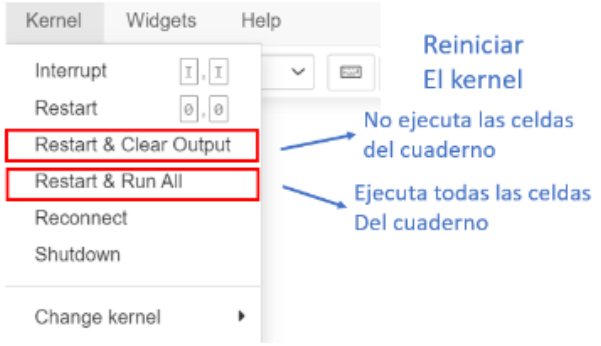
Barra con el nombre del cuaderno y última versión del cuaderno



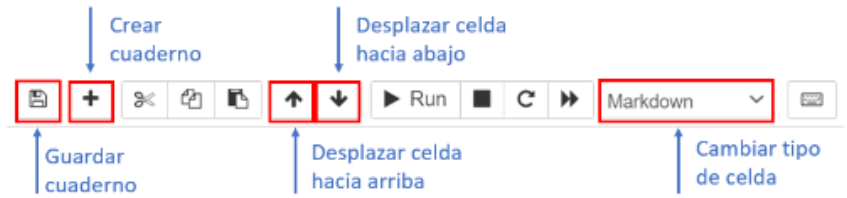
Jupyter Notebook: primeros pasos

Barra de Menu





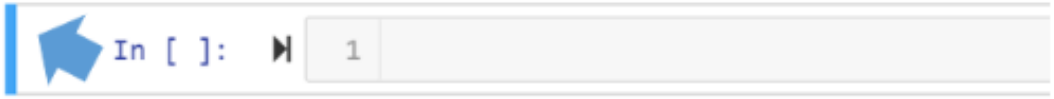
Barra de Menu gráfico



Jupyter Notebook: primeros pasos

Modos de trabajo en jupyter

- Modo Edición
  - Permite editar la celda
  - Para ejecutarla:
    - `Shift` + `ENTER`: Ejecuta el código o el markdown en una celda y coloca el **cursor** en la **siguiente celda**
    - `Ctrl` + `ENTER`: Ejecuta el código o el markdown en una celda quedando el **cursor** en la **misma celda**
  - Color verde
  - Cursor en el interior de la celda
  - 

- Modo Comando
  - Permite editar y ejecutar la celda
  - Color azul
  - Cursor a la izquierda del identificador de la celda
  - 

Jupyter Notebook: primeros pasos

Algunos atajos (shortcuts) de jupyter notebook en Modo Comando

- `m`: Cambiar celda a *Markdown*
- `y`: Cambiar celda a *Code* (Código)

- `a`: Ingresar una nueva celda por **encima** de la actual
- `b`: Ingresar una nueva celda por **debajo** de la *actual*

- `Alt` + `Enter`: Ejecutar la celda seleccionada e insertar una nueva celda.
- `Ctrl` + `s`: Guardar el documento.

- `D` + `D`: Borrar la celda
- `o`: Alternar entre ocultar y mostrar la salida

Indicadores del identificador (o cabecera de la celda)

- **In [ ]**:
  - Celda aún no se ha ejecutado
- **In [*]**:
  - Celda en ejecución
- **In [3]**:
  - Celda 3 ejecutada

Próximos pasos:

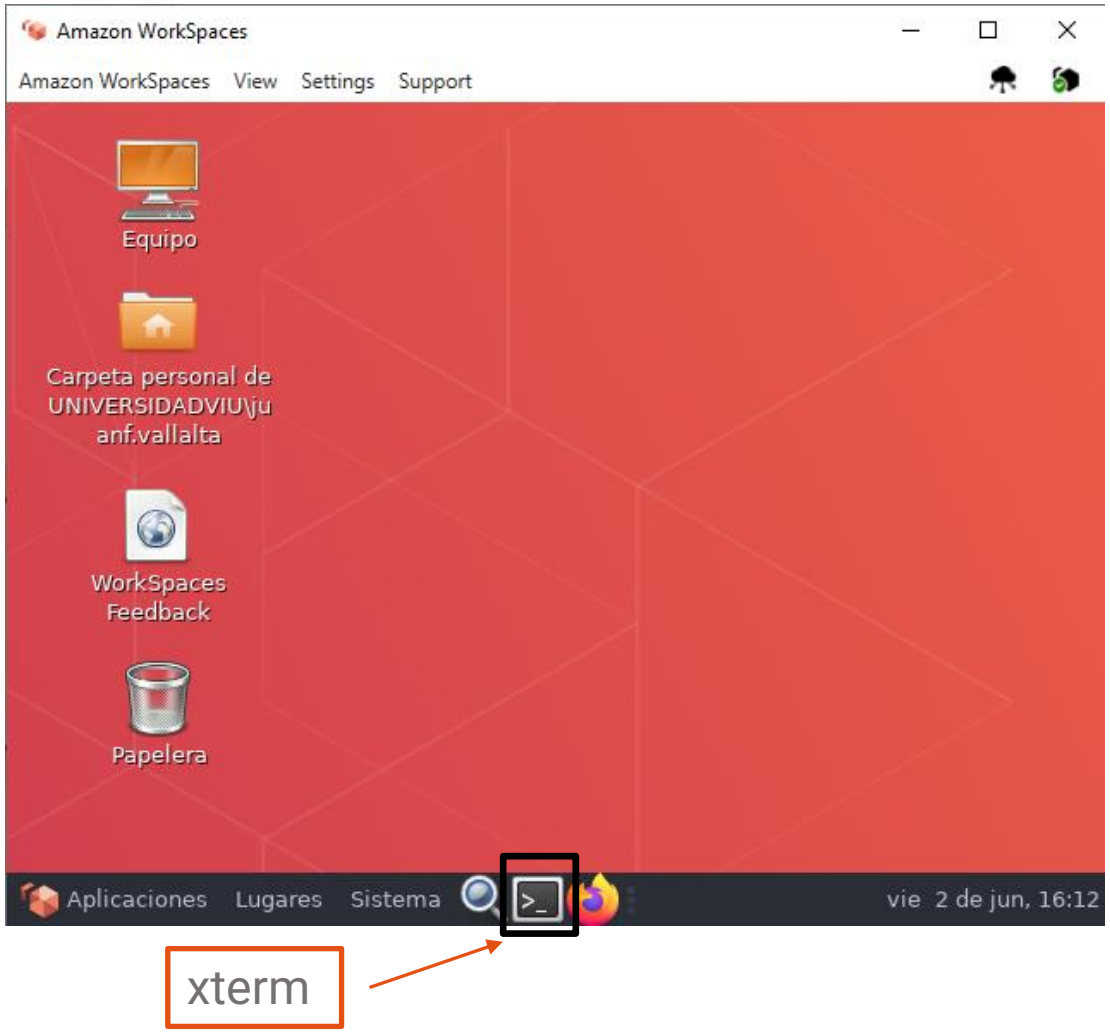
- Instalación de Conda
- Creación de un entorno virtual
- Instalar Jupyter y paquetes de Python y Jupyter Notebook.
- Abrir y ejecutar cuaderno Jupyter_Notebooks_101

Próximos pasos: instalación de Conda

- **Conda** es un **gestor de paquetes** y un **sistema de gestión de entornos**. Podemos crear un entorno en conda con todo el software que necesitamos para nuestra asignatura.

- Estos archivos pueden compartirse fácilmente con la última actualización que hayamos realizado paraque en otro lugar pueda **replicarse el mismo entorno**.

- Pasos:
  1. Ingresar a su cuenta Amazon Workspaces con sus credenciales.
  2. Desde Amazon Workspaces abrir un intérprete de comandos (xterm) (icono contiguo a la lupa).



Próximos pasos: instalación de Conda

3. Desde la consola xterm ejecutar los siguientes seis comandos de instalación:

```
mkdir -p ~/miniconda3
```

```
wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh
-O ~/miniconda3/miniconda.sh
```

```
bash ~/miniconda3/miniconda.sh -b -u -p ~/miniconda3
```

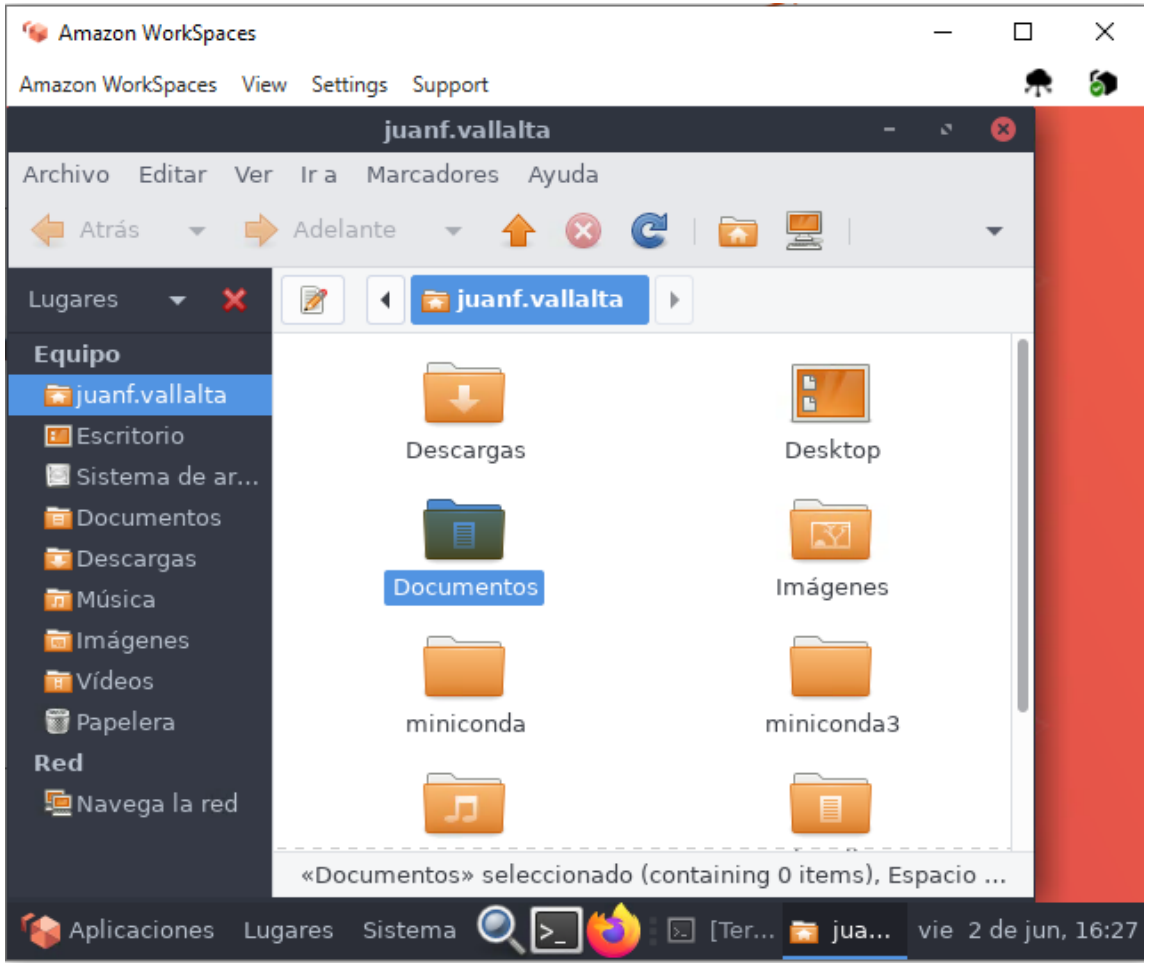
```
rm -rf ~/miniconda3/miniconda.sh
```

```
~/miniconda3/bin/conda init bash
```

Próximos pasos: instalación de Conda

4. En el directorio por defecto del usuario (Carpeta Home) ubicar el directorio **Documentos.**

5. Crear un directorio **Notebooks** con dos subdirectorios **Python** y **R**.



Próximos pasos: creación de un entorno virtual

Desde la consola **xterm** de Amazon Workspaces:

1. Crear un entorno virtual de Python.

```
conda create --name prog_python
```

Nota: Reiniciar la ventana de terminal (cerrar y abrir de nuevo)(ahora aparece)

2. Activar el entorno virtual recién creado.

```
conda activate prog_python
```

Importante tener en mente que instalaremos todos los paquetes en el entorno recién creado.
Los cuadernos también los ejecutaremos siempre desde el nuevo entorno virtual (prog_python)

Próximos pasos: instalación de Jupyter

Desde la consola **xterm** de Amazon Workspaces:

1. Instalar Jupyter.

```
conda install jupyter
```

2. Instalar los siguientes paquetes:

```
conda install pandas
```

```
conda install matplotlib
```

Instalar **biopython**

```
conda install biopython
```

Si falta algo :

```
conda update biopython
```

Próximos pasos: abrir un cuaderno de Jupyter

Desde la consola **xterm** de Amazon Workspaces:

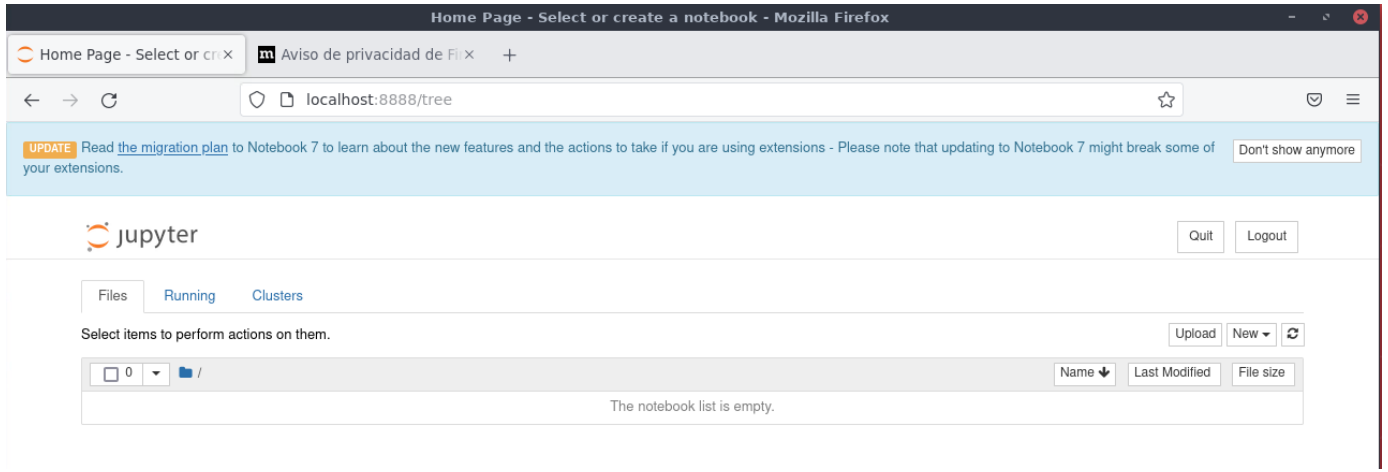
- Cambiar al directorio en que se encuentran los notebooks.

  cd Documentos/Notebooks/Python

- Ejecutar Jupyter.

  jupyter notebook

- Crear un cuaderno **prueba.ipynb**



Próximos pasos: abrir un cuaderno de Jupyter

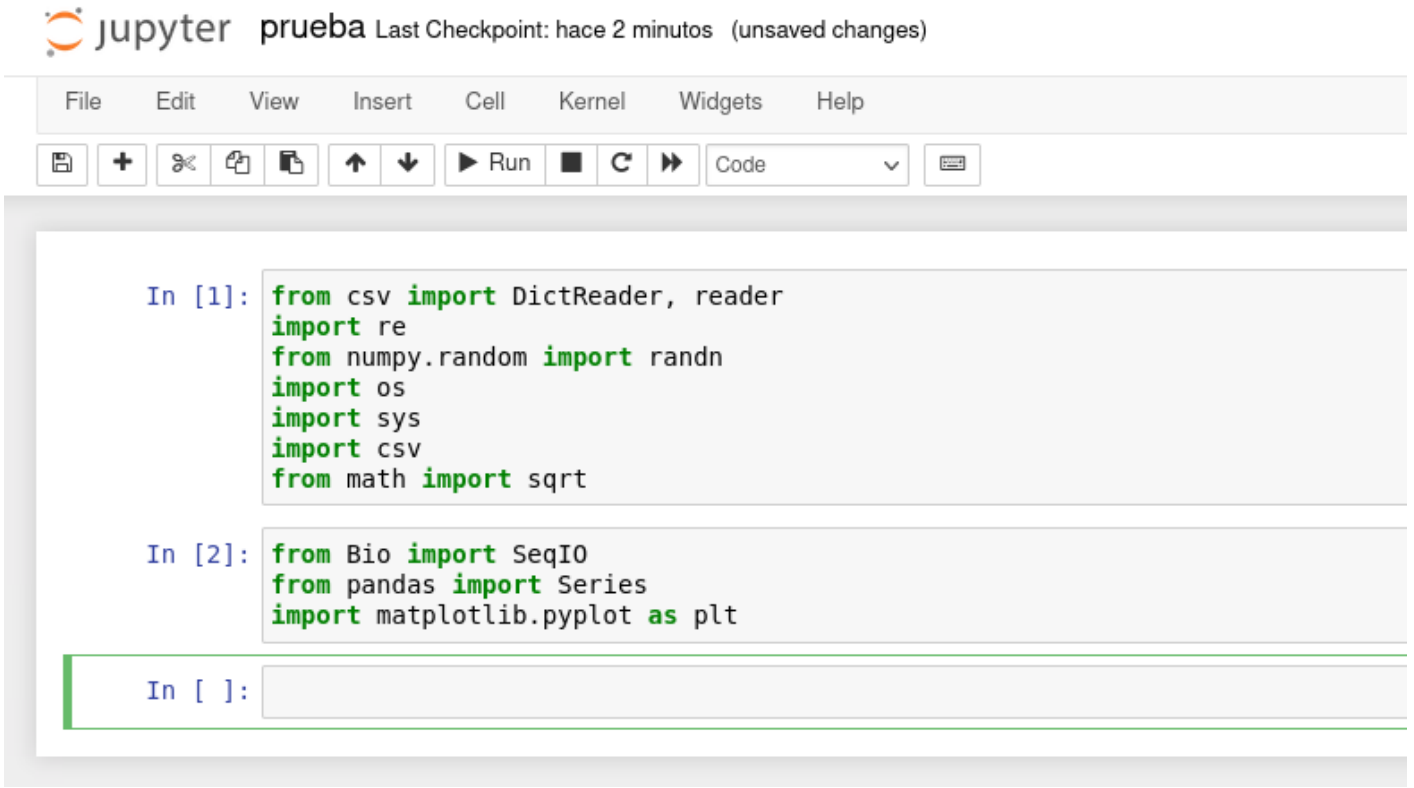
- Verificar que es posible cargar las librerías instaladas con Python: .

```
from csv import DictReader, reader
import re
from numpy.random import randn
import os
import sys
import csv
from math import sqrt
```

- Verificar que es posible cargar las librerías instaladas con conda: .

```
from Bio import SeqIO
from pandas import Series

import matplotlib.pyplot as plt
```



Próximos pasos: salir de Jupyter y cerrar el entorno de trabajo

Para salir de jupyter notebook:

a. Cerrar los cuadernos abiertos
   File → Close and Halt

b. Cerrar jupyter
   En la página del navegador en la que aparece el listado de los cuadernos y directorios:
   Pinchar el botón **Quit** y luego el botón **Logout** situados en la esquina superior derecha

Salir del entorno de trabajo

```
conda deactivate
```

```
exit
```

Resumen de comandos gestión entornos virtuales

```
conda create -n prog_python
```
crear ambiente virtual de nombre prog_python

Nota: Reiniciar la ventana de terminal (cerrar y abrir de nuevo)(ahora aparece)

Nota: Lo crea en /home/liliana.gavidia/miniconda3/envs/prog_python

```
conda activate prog_python
```
#activar ambiente virtual prog_python

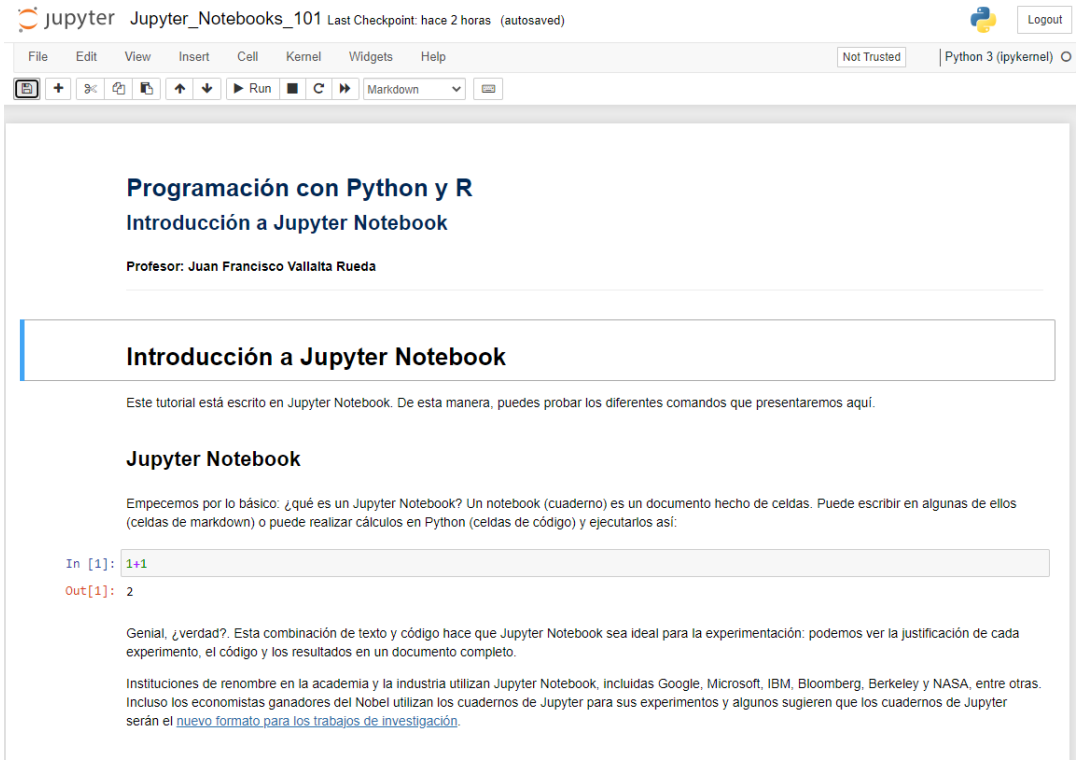
```
conda deactivate
```
comando a usar cuando desees salir de ambiente virtual

```
conda info -e
```
listar ambientes virtuales

```
conda info --envs
```
listar ambientes virtuales

Ejercicio:

- Abrir y ejecutar cuaderno Jupyter_Notebooks_101
- https://drive.google.com/file/d/1wGSJBfAmOgzXIdjsBWAPqlh8Ek9rZIDv/view?usp=sharing



Introducción a Python

02MBIF – Programación con Python y R

Universidad Internacional de Valencia

De:
Planeta Formación y Universidades