



UNIVERSIDAD
COMPLUTENSE
MADRID

Desarrollo de Aplicaciones y Sistemas Inteligentes (DASI) Presentación de la asignatura

Pablo Gervás Gómez-Navarro
Gonzalo Méndez Pozo
Juan Pavón Mestras

Dep. Ingeniería del Software e Inteligencia Artificial UCM

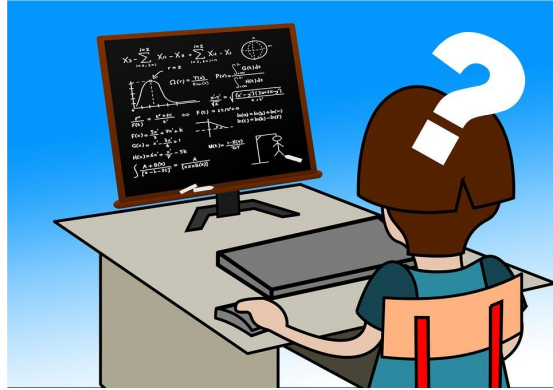
Profesores

- Pablo Gervás Gómez-Navarro - pgervas@sip.ucm.es
- Gonzalo Méndez Pozo - gmendez@fdi.ucm.es
- Juan Pavón Mestras - jpavon@fdi.ucm.es

- Página web de la asignatura:
 - En el campus virtual de la UCM (moodle):
<http://www.ucm.es/campusvirtual>

Presentaciones

- Quiénes somos
- Qué esperamos de la asignatura/máster
- Qué sabemos
 - ¿ Python ?
 - ¿ IA ?



Objetivos de la asignatura

- Aprender a construir **sistemas inteligentes**
 - Proceso de desarrollo
 - Trabajo en grupo
- Aprender a utilizar **herramientas de código abierto**
 - Aprendizaje automático
 - Sistemas multi-agentes
 - Procesamiento de lenguaje natural
 - Chatbots
- Realización de un **proyecto**
 - Aprender haciendo
 - Prototipo funcional
 - Demos a lo largo del curso

Temas

- Parte I (Lunes) - Profesor: Juan Pavón
 - Introducción a los Sistemas Inteligentes
 - Sistemas multi-agentes
 - Simulación de sistemas complejos basada en agentes
 - Aprendizaje Automático, MLOps
- Parte II (Jueves) - Profesores: Pablo Gervás y Gonzalo Méndez
 - Procesamiento de lenguaje natural
 - Agentes conversacionales: chatbots



Método de trabajo

- Clases teórico/prácticas
 - Revisión de la sesión anterior
 - Introducción a un tema
 - Prácticas con herramientas ⇒ Ejercicios evaluables
 - Trabajo en grupos
 - Desarrollo de un proyecto: Análisis, Diseño, Revisión
 - Experimentar con herramientas
 - Presentaciones de grupos
 - Seguimiento de las prácticas
 - Discusión sobre temas de la asignatura
- Trabajo fuera del aula
 - Investigar herramientas
 - Implementación
 - Preparar presentaciones



Proyecto de la asignatura

- El proyecto abordará alguno de los temas principales de la asignatura
 - Un chatbot
 - Un sistema de decisión usando alguna técnica de AA
 - Un sistema multi-agente
 - Una combinación de varias técnicas
- Proyecto en **equipos de trabajo**
 - 2-3 personas
 - Cada equipo decidirá su estructura organizativa
 - Es conveniente repartir responsabilidades
 - El grupo se autogestiona
 - Roles en cada equipo
 - Responsable/Portavoz del grupo
 - Coordinador/Documentalista de preguntas y respuestas (FAQ wiki en campus virtual)
 - Otros: diseñador, documentalista, programador, ...
 - Se podrá reestructurar el grupo en el caso de que no progrese adecuadamente



Evaluación y seguimiento de los Equipos de trabajo

- **Evaluación global de las actividades y de los resultados del grupo**
- No acabar las prácticas en un equipo implica **suspenso**
- Proyectos con resultados reproducibles
- Código en **GitHub**
- Cada equipo expondrá la evolución de su trabajo en clase periódicamente
 - Los demás grupos evaluarán
 - Se valorarán las presentaciones para la nota final
- **Seguimiento y evaluación de las actividades de cada miembro del grupo**
 - Aportaciones individuales
 - Presentaciones
 - Participación en clase



Evaluación del proyecto

- Se evaluará el proyecto desarrollado a lo largo del curso
 - Evaluación continua
- Inicialmente se prevén tres entregas:
 1. Especificación de la aplicación
 - Casos de uso, escenarios principales
 - Prototipo cableado
 - El profesor podrá proponer modificaciones en los requisitos de aspecto y funcionalidad
 2. Prototipo de caso de uso principal
 - Prototipo funcional de un escenario representativo completo
 - Arquitectura del sistema
 3. Entrega final
 - Demostración del proyecto
 - Presentación en clase
 - Documentación



Evaluación del proyecto

- Criterios a evaluar
 - Evolución del proyecto a lo largo del curso
 - Aspectos de "inteligencia" del sistema desarrollado
 - Análisis de requisitos: Casos de uso. Descripción de escenarios principales de los casos de uso
 - Diseño de organización del sistema (p.ej. agentes y recursos, componentes)
 - Modelo de información
 - Facilidad de instalación, configuración y funcionamiento
 - Prototipo funcionando
 - Conclusiones y reflexión sobre posibles líneas de trabajo futuro
- Cada miembro del grupo será evaluado por su contribución
 - No tiene por qué ser la misma nota para todos los miembros del grupo
- Algunos o todos los miembros del grupo pueden ser convocados a un examen final
 - Examen final práctico sobre el proyecto



Calificación final

- Según la ficha de la asignatura:

- Proyecto (70% de la calificación final)

- Trabajo en grupo
- Tres entregas, se califica al final teniendo en cuenta la evolución durante el curso

Proyecto * 0,7



- Ejercicios en clase (25% de la calificación final)

- Implementar programas que se plantean en clase
- Entregas por el campus virtual

Media de ejercicios *
0,25



- Asistencia con participación (5% de la calificación final)

- Participación activa y resolución de casos prácticos
- Pequeños problemas que se plantean en clase
- Contribuciones al foro del campus virtual

Participación * 0,0,5
+
Puntos extra



Bibliografía

- ANA MAS: Agentes software y sistemas multiagente: Conceptos, arquitecturas y aplicaciones, Pearson – Prentice Hall, 2005.
- Sridhar Alla & Suman Kalyan Adari: *Beginning MLOps with MLFlow. Deploy Models in AWS SageMaker, Google Cloud, and Microsoft Azure*. Apress, 2021.
- Stuart Russell, Peter Norvig: *Artificial Intelligence: A Modern Approach, 3rd edition*. Prentice Hall, 2016. Teik Toe Teoh & Zheng Rong: *Artificial Intelligence with Python*. Springer Nature, 2022.
- Yuxi (Hayden) Liu: *Python Machine Learning By Example, Third Edition*. Packt Publishing, 2020.
- Documentación en línea de las distintas herramientas

Podéis encontrar mejor bibliografía y publicarla en el campus virtual



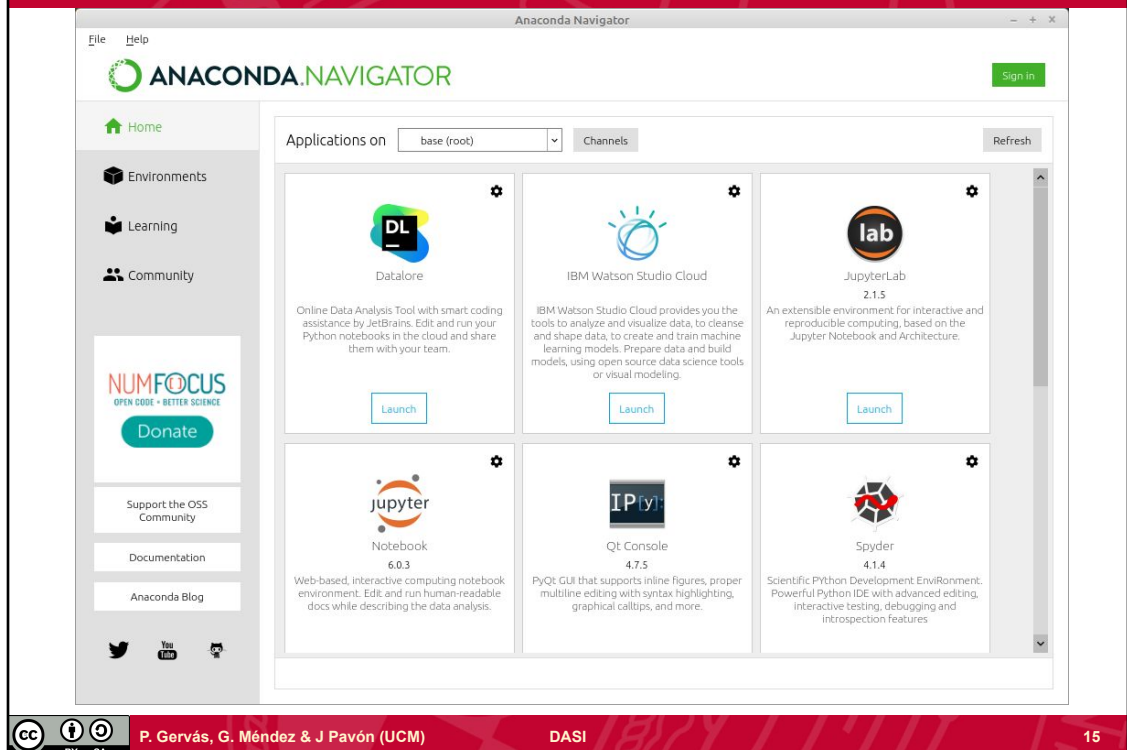
⇒ Traer instalado Anaconda en el portátil

Anaconda

- Para trabajar con distintas librerías de Python, instalar Anaconda
 - <https://www.anaconda.com/>
 - Descargar **Anaconda Distribution**
 - Más de 7500 paquetes (la mayoría open source)
 - Más de 600 megas
 - Al instalar
 - Se instalan unos 250 paquetes
 - Se pueden instalar más con `conda install`
- ⇒ Anaconda se encarga de gestionar todas las dependencias de los distintos paquetes
- Probar Anaconda
 - Arrancar el navegador de Anaconda: `anaconda-navigator`



Anaconda Navigator

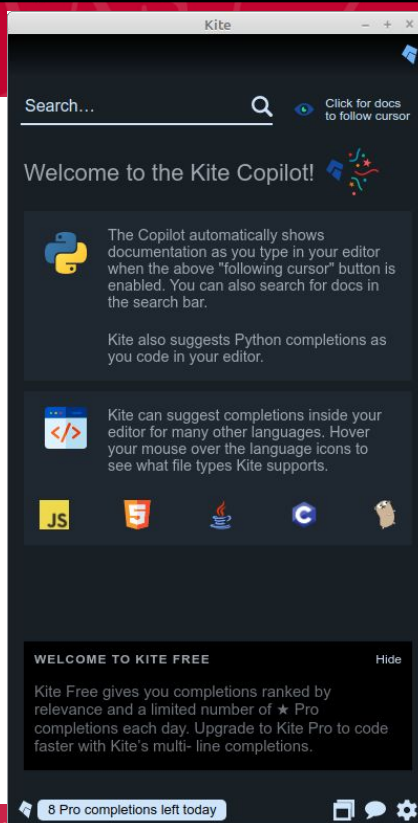
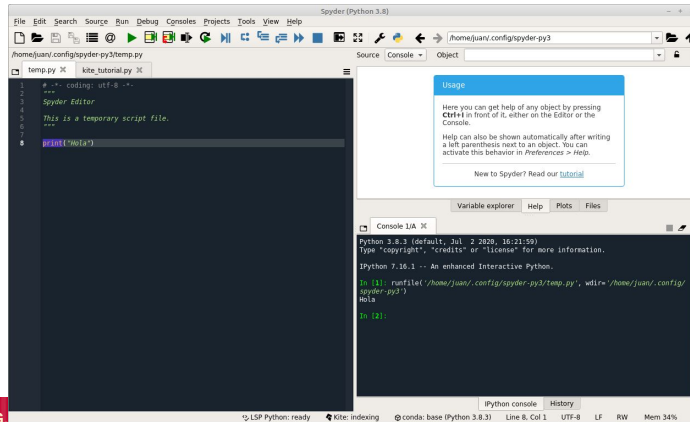


Anaconda Navigator

- El manual de usuario de Anaconda y cómo empezar a usarlo:
<https://docs.anaconda.com/anaconda/user-guide/getting-started/>
- Al arrancar el navegador de Anaconda aparecen varios editores:
 - **Spyder** ⇒ entorno de desarrollo y depuración (IDE)
 - **Jupyter Notebook** ⇒ IDE integrado en el navegador
 - También están instalados otros, como **PyCharm**
- Y algunas herramientas para tareas específicas:
 - **IBM Watson Studio cloud**: Plataforma de IBM para aprendizaje máquina
 - **RStudio**: IDE para R (herramientas estadísticas y gráficos)
 - Orange 3, GlueVix, ... para tratamiento y visualización de datos

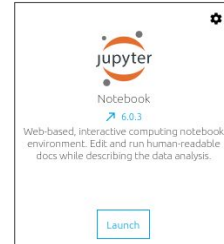


- Entorno de desarrollo integrado (IDE)
 - Editor de programas avanzado
 - Con el plugin Kite mejora la ayuda para completar código y documentación de las funciones
 - Consola de ejecución IPython (interactive Python)
 - Explorador de variables (visualizar, modificar, etc.)
 - Plot (para ver y copiar imágenes de resultados de ejecución)
 - Depurador de programas





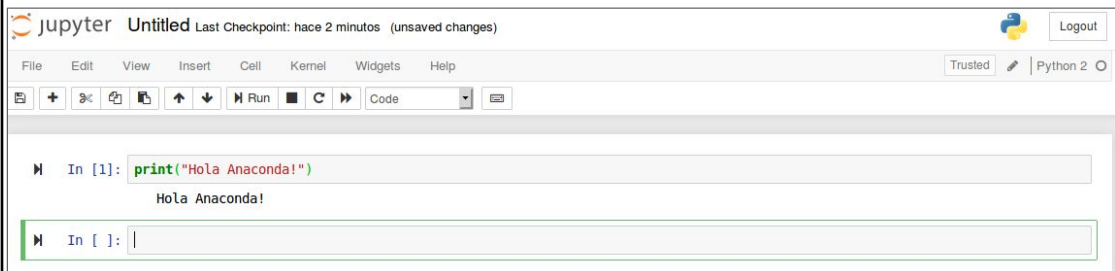
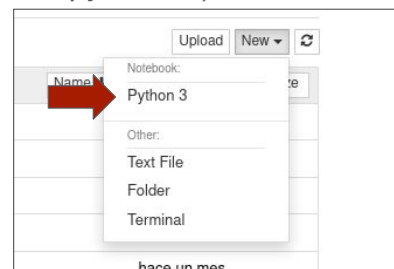
El Notebook de Jupyter

- Es un entorno para desarrollar programas Python pero también documentación: código vivo (se puede ejecutar), texto, fórmulas, figuras, y medios audiovisuales
 - Se visualiza en un navegador (Chrome, Firefox, Opera, etc.)
 - Es útil para análisis de datos con Python: importación y exportación, manipulación y transformación, visualización, etc.
- Se puede lanzar
 - Desde la consola de anaconda =====>
 - o desde la línea de comandos
(desde el directorio de trabajo correspondiente):
`directorio-de-trabajo> jupyter notebook`
- Se arranca un servidor de Jupyter
 - y en el navegador aparecerá una ventana con el notebook y listado del directorio de trabajo




El cuaderno (notebook) de Jupyter

- Arranca **Jupyter Notebook** (no confundir con *Jupyter Lab*)
 - Seleccionar **New** → **Notebook: Python 3**
 - En la primera línea escribir:
`print("Hola Anaconda!")`
 - Salvar pulsando el botón 
 - Ejecutar con Run 




Tu primer programa: Hello World!

- El más famoso del mundo: <https://helloworldcollection.github.io/>
 - En más de 570 lenguajes de programación
- En el notebook escribe:


```
print("Hello, World!")
```
- Y ejecuta este código pulsando el botón 

Los cuadernos (*notebooks*) de Jupyter

- Un *notebook* de Jupyter es un fichero
 - Con extensión **.ipyb**
 - Consta de un conjunto de **celdas**
 - Son de varios tipos y se pueden combinar en un notebook
 - Se añaden con el botón **+**
- Tipos de celdas:
 - **Markdown**: Para escribir texto formateado
<https://daringfireball.net/projects/markdown/>
 - **Code**: Código ejecutable
 - Marcadas por la palabra **In [n]** para el código
 - Marcadas por la palabra **Out[n]**, para el resultado de ejecución

⇒ Para ejecutar el código, pulsar el botón de ejecución 
 - **Raw NBConvert**: Código no ejecutable

Guardando los notebooks como ficheros

- El código de los notebooks se puede guardar para luego ejecutarlos
 - Pulsando el botón 
 - O bien con:
File → Save as...
- También se puede guardar como ficheros puramente Python (.py)
 - Guarda el fichero con **Download as → Python (.py)**
 - En principio se guardará como **Untitled1.py**
- Se podría cambiar el nombre del notebook con: **File → Rename**
 - o clickando en el nombre del notebook:



El cuaderno (notebook) de Jupyter - Ejercicio

- Prueba a crear un notebook, añade varias celdas de código, y ejecútalas. Por ejemplo:

```
# Código sencillo:
print ("hola")

# Código que genera una gráfica
import scipy.special as spec
%pylab inline
x = np.linspace(0, 20, 200)
for n in range(0,13,3):
    plt.plot(x, spec.jn(n, x), label=r'$J_{%i}(x)$' % n)
grid()
legend()
title('Bessel Functions are neat');

# Añade un vídeo de youtube
from IPython.display import YouTubeVideo
YouTubeVideo('A4B-Qr0voyY')
```

- Más ejemplos en el tutorial de Jupyter:

<http://nbviewer.jupyter.org/github/ipython/ipython/blob/1.x/examples/notebooks/Part%20-%20Rich%20Display%20System.ipynb>