Hoja de trucos del Módulo 5: Opciones de Entorno de Desarrollo y Ejecución

Paquete o método	Descripción	Ejemplo de código

SparkSession.builder.getOrCreate()	Obtiene un SparkSession existente o, si no hay ninguno, crea uno nuevo basado en las opciones establecidas en este constructor.	`from pyspark.sql import SparkSession`
		
		Crea un SparkSession o obtiene uno existente:
		`spark = SparkSession.builder.appName("myApp").getOrCreate()`
		
		Ahora puedes usar el objeto spark para trabajar con Spark

cd	Se utiliza para moverse eficientemente desde el directorio de trabajo actual a diferentes directorios en tu sistema.	Sintaxis básica del comando cd:
		`cd [opciones]... [directorio]`
		
		Ejemplo 1: Cambiar la ubicación del directorio a folder1:
		`cd /usr/local/folder1`
		
		Ejemplo 2: Regresar al directorio de trabajo anterior
		`cd -`
		
		Ejemplo 3: Subir un nivel desde el directorio de trabajo actual
		`cd ..`

Paquete o método	Descripción	Ejemplo de código

docker-compose	Herramienta para definir y ejecutar aplicaciones Docker multicontenedor. Utiliza un archivo YAML para configurar los servicios y nos permite crear e iniciar todos los servicios desde un solo archivo de configuración.	Ejemplo (docker-compose.yml)
		`version: '3'`
		`services:`
		`  web:`
		`  image: nginx:latest`
		`  ports:`
		`  - "80:80"`
		`  db:`
		`  image: postgres:latest`

git clone	Puedes crear una copia de un repositorio específico o de una rama dentro de un repositorio.	`git clone REPOSITORY_URL [DIRECTORIO_DESTINO]`
import	Se utiliza para hacer que el código de un módulo sea accesible en otro. Las importaciones en Python son cruciales para una estructura de código exitosa. Puedes reutilizar código y mantener tus proyectos manejables utilizando importaciones de manera efectiva para aumentar la productividad.	`from pyspark.sql import SparkSession`
pip	Para asegurar que las solicitudes funcionen, el programa pip busca el paquete en el Índice de Paquetes de Python (PyPI), resuelve cualquier dependencia e instala todo en tu entorno Python actual.	`pip list`
pip install	El comando pip install <paquete> busca la última versión del paquete e instala.	`pip install nombre_del_paquete`
pip3 install	pip3 es el administrador de paquetes oficial y el comando pip para Python 3. Permite instalar y gestionar paquetes de software de terceros con características y funcionalidades que no se encuentran en la biblioteca estándar de Python. Pip3 instala paquetes desde PyPI (Índice de Paquetes de Python), pero no resolverá dependencias ni te ayudará a resolver conflictos de dependencias.	`pip3 install nombre_del_paquete`
print()	Imprime el mensaje especificado en la pantalla u otro dispositivo de salida estándar.	`print("¡Hola, Mundo!")`

Paquete o método	Descripción	Ejemplo de código
	El mensaje puede ser una cadena o cualquier otro objeto; el objeto se convertirá en una cadena antes de ser escrito en la pantalla.	

python3	Python 3 es un lenguaje de programación ampliamente utilizado, conocido por su legibilidad y versatilidad.	`sudo apt-get update`
		`sudo apt-get install python3`
		`python3 --version #para verificar la versión de python`

sc.setloglevel()	Usando este método, puedes cambiar el nivel de registro al nivel deseado. Los niveles de registro válidos incluyen ALL, DEBUG, ERROR, FATAL, INFO, OFF, TRACE y WARN.	Importar los módulos necesarios:
		`from pyspark import SparkContext`
		Crear un SparkContext:
		`sc = SparkContext("local", "LogLevelExample")`
		Establecer el nivel de registro a un nivel deseado (por ejemplo, INFO, ERROR):
		`sc.setLogLevel("INFO")`
		Ahora, cualquier mensaje de registro con un nivel de gravedad igual o mayor que INFO se mostrará

setMaster()	Indica dónde ejecutar tu aplicación Spark, local o en clúster. Cuando ejecutas en un clúster, necesitas especificar la dirección del maestro de Spark o la URL del Driver para un clúster distribuido. Normalmente asignamos un valor local[*] a setMaster() en Spark mientras hacemos pruebas internas.	`<pre>from pyspark import SparkContext </pre>`
		`<p>Crear un SparkContext con un maestro local:</p>`
		`<pre>sc = SparkContext("local[*]", "MyApp")</pre></td>`

show()	El método show() de Spark DataFrame se utiliza para mostrar el contenido del DataFrame en un formato de tabla con filas y columnas. Por defecto, muestra solo veinte filas, y los valores de las columnas se truncarán a veinte caracteres.	`df.show()`

Paquete o método	Descripción	Ejemplo de código
source	Se utiliza para ejecutar un archivo de script en el entorno de shell actual, permitiéndote modificar el entorno de shell actual de la misma manera que lo harías si hubieras escrito comandos manualmente.	Asumiendo un shell Bash: `source myscript.sh`
virtualenv	Principalmente una aplicación de línea de comandos. Modifica las variables de entorno en un shell para crear un entorno Python aislado, por lo que necesitarás tener un shell para ejecutarlo. Puedes escribir virtualenv (nombre de la aplicación), seguido de banderas que controlan su comportamiento.	Crear un entorno virtual llamado "myenv": `virtualenv myenv`