Python 3 para impacientes

"Simple es mejor que complejo" (Tim Peters)

martes, 26 de agosto de 2014

Administrar sistemas con IPython

Una de la características importantes que tiene la consola IPython es su capacidad de ejecutar, además de código Python, comandos del sistema operativo, programas externos y scripts escritos en otros lenguajes; evitándonos tener que utilizar otras consolas al mismo tiempo

Además de lo comentado, el hecho de permitir personalizar ampliamente su funcionamiento convierte a IPython en una herramienta interesante para cualquier persona que administre los sistemas de una empresa.

En ese caso el técnico contará con la inestimable ayuda de las funciones mágicas de IPython que podrán crecer en número con el transcurso del tiempo. También, probablemente, será recurrente el uso de la función mágica **%alias** para acortar los comandos más utilizados y el desarrollo y ejecución de scripts propios que resuelvan las tareas más comunes.

Un caso práctico de personalización básica

Para personalizar el entorno de trabajo podemos utilizar los distintos archivos de configuración que se guardan en un perfil IPython e incorporar uno o más archivos al directorio "***startup***" (que se encuentra también dentro del perfil) para que se ejecuten antes de comenzar una sesión de trabajo.

- Ejemplo de archivo "**ipython_config.py**":

```
c = get_config()

c.TerminalIPythonApp.display_banner = False
c.InteractiveShellApp.hide_initial_ns = False
```

En este ejemplo básico de archivo "**ipython_config.py**" se desactiva la visualización de la ayuda que se muestra al principio de una sesión IPython (display_banner = False) y si se declaran variables en un archivo ejecutable de la carpeta "**startup**" estarán visibles y podrán utilizarse libremente en nuestros comandos o scripts (hide_initial_ns = False).

(En el sitio oficial de IPython podrá obtenerse ayuda de muchas opciones de configuración del archivo "ipython_config.py" y de otros archivos que configuran la consola QT y Notebook).

Buscar

Buscar

Python para impacientes

Python
IPython
EasyGUI
Tkinter
JupyterLab
Numpy

Anexos

Guía urgente de MySQL
Guía rápida de SQLite3

Entradas + populares

Dar color a las salidas en la consola
En Python para dar color a las salidas en la consola (o en la terminal de texto) existen varias posibilidades. Hay un método basado ...

Instalación de Python, paso a paso
Instalación de Python 3.6 A finales de 2016 se produjo el lanzamiento de Python 3.6 . El propósito de esta entrada es mostrar, pas...

Variables de control en Tkinter
Variables de control Las variables de control son objetos especiales que se asocian a los widgets para almacenar sus valore...

Añadir, consultar, modificar y suprimir elementos en Numpy
Acceder a los elementos de un array. [], [,], ... Acceder a un elemento de un array. Para acceder a un elemento se utiliz...

Cálculo con arrays Numpy
Numpy ofrece todo lo necesario para obtener un buen rendimiento cuando se trata de hacer cálculos con arrays. Por como está concebid...

Operaciones con fechas y horas. Calendarios
Los módulos datetime y calendar amplían las posibilidades del módulo time que provee funciones para manipular expresiones de ti...

Tkinter: interfaces gráficas en Python
Introducción Con Python hay muchas posibilidades para programar una interfaz gráfica de usuario (GUI) pero Tkinter es fácil d...

Convertir, copiar, ordenar, unir y dividir arrays Numpy
Esta entrada trata sobre algunos métodos que se utilizan en Numpy para convertir listas en arrays y viceversa; para copiar arrays d...

Tkinter: Tipos de ventanas
Ventanas de aplicación y de diálogos En la entrada anterior tratamos los distintos gestores de geometría que se utilizan para di...

El módulo random
El módulo random de la librería estándar de Python incluye un conjunto de funciones

- Ejemplo de archivo ejecutable "**inicio.ipy**" para la carpeta "**startup**":

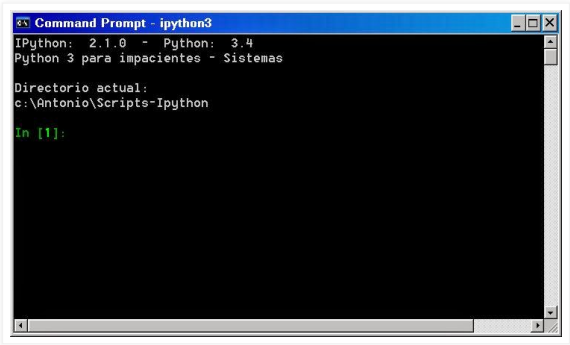
```
# coding: utf-8

# Carga de módulos

import IPython
import sys

# Declaración de variables

dominio = "and.empresa.es"
equipo = "eq99999"
usuario = "us123"
grupo = "servidores"
mensaje = "mensaje"
sesion = "1"

# Personalización de la pantalla inicial

get_ipython().magic('%cls')
print("IPython: ", IPython.__version__, " - ", "Python: ", sys.winver)
print("Python 3 para impacientes - Sistemas\n")
print("Directorio actual: ")
get_ipython().magic('%alias scripts dir *.py *.cmd')
get_ipython().magic('%cd c:\\usuario/Scripts-Ipython/')
get_ipython().magic('%pwd')
```

Al principio del archivo "**inicio.ypi**" se importan los módulos **IPython** y **sys**; después se declaran algunas variables y, finalmente, se personaliza la pantalla inicial de la consola. Para esto último se realiza antes un borrado de la pantalla y a continuación se muestran, a título informativo, las versiones instaladas de Python y IPython.

Después, se define el alias "**scripts**" que listará, cuando lo ejecutemos, los archivos .py/.cmd; y para finalizar se cambia el directorio de trabajo predeterminado y se muestra su ruta en pantalla.

Ejecución de comandos y scripts

A partir de este momento cada vez que iniciemos una sesión de trabajo con la consola IPython se interpretarán las opciones del archivo de configuración "**ipython_config.py**" y se ejecutarán las líneas del archivo "**inicio.ipy**", visualizándose en pantalla la siguiente información:



La consola quedará lista para ejecutar cualquier comando o script que precisemos que, además, podrán incluir las variables que se declararon en el archivo "**inicio.ipy**". Para mostrar las variables utilizaremos las funciones mágicas **%who** y **%whos**: la primera presentará la lista de variables disponibles y **%whos** ampliará la información mostrando el tipo y el valor de cada una de ellas:

Ejemplos de comandos que usan las variables de inicio.ipy

1) Mostrar información de la memoria del equipo "servidor01":

: equipo = "servidor01"
: !systeminfo /S $equipo | grep "Memory"

2) Mostrar las sesiones de Escritorio Remoto abiertas en "servidor02":

: equipo = "servidor02"
: !qwinsta /SERVER:$equipo

que permiten obtener de distintos modos números a...

Archivo

agosto 2014 (15)

python.org

pypi.org

Sitios

- ActivePython
- Anaconda
- Bpython
- Django
- Flask
- Ipython
- IronPython
- Matplotlib
- MicroPython
- Numpy
- Pandas
- Pillow
- PortablePython
- PyBrain
- PyCharm
- PyDev
- PyGame
- Pypi
- PyPy
- Pyramid
- Python.org
- PyTorch
- SciPy.org
- Spyder
- Tensorflow
- TurboGears

3) Mostrar información de Active Directory de la cuenta de equipo "cd01":

: equipo = "cd01"
: !dsquery computer -name $equipo -domain $dominio | dsget computer

En este caso se utiliza la variable $dominio con el valor por defecto.

4) Listar software del equipo anterior:

: !wmic /node:$equipo product get name,version

Ejemplos de scripts que utilizan las variables de inicio.ipy

1) Script para mostrar grupos del usuario "usuario1" de Active Directory:

(grupos.py):

```
get_ipython().system('echo Grupos:')

info='echo dsquery user -name $usuario -domain $dominio -limit 0'
info+= '; dsget user -memberof -L'
get_ipython().system(info)

comando='dsquery user -name $usuario -domain $dominio -limit 0'
comando+=' | dsget user -memberof -L'
get_ipython().system(comando)
```

Para ejecutar el script:

: usuario = "usuario01"
: run grupos.py

2) Enviar un mensaje a todos los usuarios conectados al servidor "aplicacion01":

(mensaje.py):

```
get_ipython().system('echo Enviar mensaje a usuarios conectados:')
get_ipython().system('echo msg * /SERVER:$equipo /V $mensaje')
get_ipython().system('msg * /SERVER:$equipo /V $mensaje')
get_ipython().system('echo.')
```

Para ejecutar el script:

: equipo = "aplicacion01"
: mensaje = "El servidor se apagará en 30 m."
: run mensaje.py

Ejecutar scripts en otros lenguajes

IPython cuenta con funciones mágicas orientadas a celdas que permiten la escritura y ejecución de scripts en otros lenguajes como Perl, Ruby, Javascript, etc.

1) El siguiente ejemplo escrito en lenguaje Perl lee todas las líneas que contiene un archivo de texto y las muestra en pantalla.

: %%perl
: $fich = "archivo.txt";
: open(DATA, $fich) || die "no se puede abrir";
: @lineas = <DATA>;
: print @lineas;
: close(DATA);
:

2) Para concluir el script Perl que sigue lee las líneas del archivo "entrada.txt" y si comienzan por la palabra "ERROR" las imprime por pantalla.

: %%perl
: my $ent="entrada.txt";
: open (ENTRADA,"<$ent") || die "no se puede abrir";
: while (<ENTRADA>)

```
: {
: print if /^ERROR/;
: }
: close (ENTRADA);
:
```

Ir al índice del tutorial de IPython

Publicado por Pherkad en 11:07

Etiquetas: IPython, Jupyter

Entrada más reciente Inicio Entrada antigua