

Introducción a la Programación con **PYTHON**



José Rodríguez

Unidad 6 - Módulos

1.- Concepto de Módulo

Para facilitar el mantenimiento y la lectura los programas demasiado largos pueden dividirse en módulos, agrupando elementos relacionados. Los módulos son entidades que permiten una organización y división lógica de nuestro código. Los ficheros son su contrapartida física: cada archivo Python almacenado en disco equivale a un módulo.

Vamos a crear nuestro primer módulo entonces creando un pequeño archivo `moduloprint.py` con el siguiente contenido:

```
def mi_funcion():  
    print "Hola mundo"
```

Si quisiéramos utilizar la funcionalidad definida en este módulo en nuestro programa tendríamos que importarlo. Para importar un módulo se utiliza la palabra clave `import` seguida del nombre del módulo, que consiste en el nombre del archivo menos la extensión.

```
import moduloprint  
moduloprint.mi_funcion()
```

El `import` no solo hace que tengamos disponible todo lo definido dentro del módulo, sino que también ejecuta el código del módulo. Por esta razón, si redefinimos el módulo de esta forma:

```
def mi_funcion():  
    print "Hola mundo"  
print "Esto es un modulo"
```

Al llamar al módulo se escribirá "Esto es un módulo".

La cláusula `import` también permite importar varios módulos en la misma línea. En el siguiente ejemplo podemos ver cómo se importa con una sola cláusula `import` los módulos de la distribución por defecto de Python `os`, que engloba funcionalidad relativa al sistema operativo; `sys`, con funcionalidad relacionada con el propio intérprete de Python y `time`, en el que se almacenan funciones para manipular fechas y horas.

```
import os, sys, time  
print time.asctime()
```

Cuando se importa un módulo se carga en memoria todo las funciones y clases de este módulo y tendremos que nombrarlas indicando el nombre del módulo seguido de un punto y el nombre de la función o clase

```
Nombremodulo.nombrefuncion()
```

Sin embargo es posible utilizar la construcción from-import para ahorrarnos el tener que indicar el nombre del módulo antes del objeto que nos interesa. De esta forma se importa el objeto o los objetos que indiquemos al espacio de nombres actual.

```
from time import asctime
print asctime()
```

A la hora de importar un módulo Python recorre todos los directorios indicados en la variable de entorno PYTHONPATH en busca de un archivo con el nombre adecuado. El valor de la variable PYTHONPATH se puede consultar desde Python mediante sys.path

```
import sys
print sys.path
```

Este script nos mostrará todos los directorios en los cuales buscará python los módulos. Si queremos añadir algún directorio nuevo tendremos que incluirlo en la variable de entorno PYTHONPATH.

2.- Interfaz con el sistema operativo

El módulo os proporciona funciones para interactuar con el sistema operativo:

```
import os
```

Para conocer las funciones incluidas en este módulo utilizamos el comando dir(os)

```
print dir(os)
```

aunque tambien podemos devolver el resultado en una lista y manipular la lista.

```
lista = dir(os)
print lista[3]
```

Si queremos ver todos los elementos de la lista la recorremos en un bucle

```
import os
lista= dir(os)
for x in lista:
    print x
```

Para obtener información detallada utilizamos la función help
Print help(os)

Esto nos mostrará una información detalla de todos los elementos que forman el módulo os. Para obtener información de un solo elemento

```
print help(os.system)
```

La respuesta sería

Help on built-in function system in module nt:

```
system(...)
    system(command) -> exit_status
    Execute the command (a string) in a subshell.
None
```

Veamos ahora algunas de las funciones incluidas en este paquete:

Como hemos visto, system permite ejecutar cualquier comando del sistema operativo sobre el que actúa. En el caso de Windows podemos utilizar cualquier comando que permita la shell en modo texto CMD.

```
import os
os.system("dir")
os.system("mkdir ejemplos")
```

Incluso podemos lanzar ejecutables como el block de notas

```
os.system("notepad.exe")
```

Algunas de las funciones relativas a ficheros las mostramos en la siguiente lista de ejemplos:

```
os.rename("1.py", "10.py")
os.rmdir("ejemplos")
```