Curso Básico de Python

10. Bibliotecas

Contenido

Python cuenta con una gran cantidad de recursos y herramientas para las más diversas áreas. En este capítulo presentamos una breve introducción a las bibliotecas disponibles más usuales.

Lecturas recomendadas:

1. Pequeño paseo por la biblitoeca estándar en el Tutorial de Python.
2. Pequeño paseo por la biblitoeca estándar - parte II en el Tutorial de Python.

1 Bibliotecas

Una *biblioteca* es una colección de implementaciones de comportamiento (programas) que pueden ser invocados a través de una interfaz sin necesidad de conocer su estructura interna. La invocación suele realizarse mediante llamadas a funciones.

El código en una biblioteca está organizado para ser utilizado por muchos programas diferentes sin relación entre sí. Esto permite la reutilización de elementos de programación estandarizados, sin tener que reiterar la implementación.

La caracteristica esencial de una biblioteca es haber sido diseñada para permitir la reutilización del código conociendo solo la interfaz, es decir la forma de invocar las funciones incluidas en la biblioteca. Python dispone de una biblioteca estándar, pero existen muchas bibliotecas para aplicaciones específicas.

2 La biblioteca estándar

En los siguientes apartados mencionamos los módulos más usuales de la biblioteca estándar, con algunos ejemplos de su ustilizacion.

Interfaz con el sistema operativo: `os`

El módulo `os` permite el acceso a comandos del sistema operativo. Usar este módulo con `import os` y calificar todas sus funciones, para evitar choques con funciones integradas (*buil-in functions*) del mismo nombre. Por ejemplo, `os.open()` opera distinto que la función estándar `open()`. Algunas funciones usuales:

`os.getcwd()` : devuelve el directorio corriente.
`os.chdir()` : cambia el directorio de trabajo
`os.chmod()` : cambia permisos de archivos y directorios.
`os.chown()` : cambia el dueño y grupo de archivos y directorios.
`os.getcwd()` : devuelve el directorio actual.
`os.listdir()` : lista el contenido de un directorio.
`os.mkdir()` : crea un directorio.
`os.remove()` : borra un archivo.
`os.rename()` : cambia de nombre un archivo o directorio.
`os.rmdir()` : borra un directorio.
`os.sync()` : vacía los buffers, sincroniza dispositivos.
`os.uname()` : informacion sobre el equipo, sistema operativo y versiones.

El módulo `shutil` provee funciones de para copiar y manejar colecciones de archivos y directorios, así como copiar y borrar.

Referencias:
- Python docs. Módulo `os`, Miscellaneous operating system interfaces.
- Python docs. Módulo `shutil`, High-level file operations.

Variables y funciones relacionadas con el intérprete: `sys`

El módulo `sys` provee acceso a variables usadas por el intérprete de comandos, así como funciones que interactúan directamente con el intérprete. Algunas variables y funciones usuales:
`sys.argv` : lista de parámetros de invocación; `sys.argv[0]` es el nombre del programa.
`sys.exit()` : termina la ejecución.
`sys.path` : lista de rutas de búsqueda.

Los siguientes objetos de tipo archivo manejan la entrada y salida de los programas:
`sys.stdin` : entrada estándar, usado por `input()`.
`sys.stdout` : salida estándar, useado por `print()`.
`sys.stderr` : error estándar, usado para mensajes de error.
Asignando estos objetos a otros archivos u objetos que se comporten como archivos, es posible redirigir toda la entrada y salida del intérprete.

Referencia:
- Python docs. Módulo `sys`, System-specific parameters and functions.

Otros módulos de la biblioteca estándar

La siguiente lista indica el contenido de algunos módulos usuales de la biblioteca estándar.

Programación en Python.
`argparse` : manejo de argumentos en línea de comando.
`builtins` : el espacio de nombres de las funciones integradas.
`glob` : uso de comodines para rutas y nombres de archivos.
`keyword` : lista de palabras clave, permite ver si una palabra es palabra clave.
`os` : interfaz a comandos del sistema operativo.
`subprocess` : corre procesos, conecta a entrada y salida estándar, obtiene códigos de retorno.
`sys` : acceso a variables y funciones del intérprete de comandos.
`threading` : ejecución paralela (hilos).
`types` : nombres de los tipos internos.

`unittest` : marco de desarrollo para pruebas de unidad (unit-testing).

Fecha y hora
`datetime` : tipos básicos de fecha y hora.
`time` : acceso a hora y conversiones.

Bases de datos, planillas.
`csv` : leer y escribir datos tabulados por delimitadores (CSV).
`sqlite3` : implementación de base de datos usando SQLite 3.x.

Internet, sitios web, correo electrónico.
`email` : manejo de mensajes de error.
`html` : utilitarios para manejar HTML.
`http` : códigos de estado y mensajes HTTP.
`ipaddress` : manejo de direcciones IPv4 e IPv6.
`xml` : módulos para procesamiento de XML.

Matemáticas, procesamiento de datos, iteraciones.
`itertools` : funciones para crear iteradores para recorrido eficiente de lazos repetitivos.
`math` : funciones matemáticas.
`random` : números pseudo aleatorios en varias distribuciones estadísticas usuales.
`statistics` : funciones estadísticas.

Serialización y compresión de datos.
`pickle` : serializa y deserializa objetos Python.
`zipfile` : leer y escribir archivos en formato ZIP.

Cadenas de caracteres.
`re` : operaciones con expresiones regulares para parocesamiento de cadenas de caracteres.
`string` : operaciones sobre cadenas de caracteres.

Interfaz gráfica.
`tkinter` : interfaz hacia Tcl/Tk para crear interfaces gráficas de usuario (GUI).

Referencia:

- Python docs. Python Module Index. Documentació de los módulos que componen la Biblioteca Estándar de Python.

3 Programación numérica

Informalmente, la programación numérica refiere a un área de las ciencias de la computación y la matemática donde se tratan algoritmos para la aproximación numérica, esencialmente algoritmos donde se resuelven problemas con variables continuas para aplicaciones en ciencia e ingeniería.

Como lenguaje de programación de propósito general, Python por sí solo no tiene la capacidade suficiente para el manejo de datos ni la velocidad requerida en la programación numérica. No obstante, con el complemento de los módulos específicos NumPy, SciPy, Matplotlib y Pandas, adquiere capacidades comparables a las de lenguajes específicos como Matlab y R.

- **Numpy** provee estructuras básicas, tales como arreglos multidimensionales y matrices.
- **SciPy** usa las estructuras de NumPy, extendiendo sus capacidades con funciones para minimización, regresión, transformadas de Fourier y muchas otras.

- **Matplotlib** es una biblioteca gráfica para Python, NumPy y SciPy.
- **Pandas** usa todos los módulos anteriores para el manejo de datos y su análisis, ofreciendo estructuras de datos y operaciones para manejar tablas numéricas y series de tiempo; su nombre deriva de "panel data".

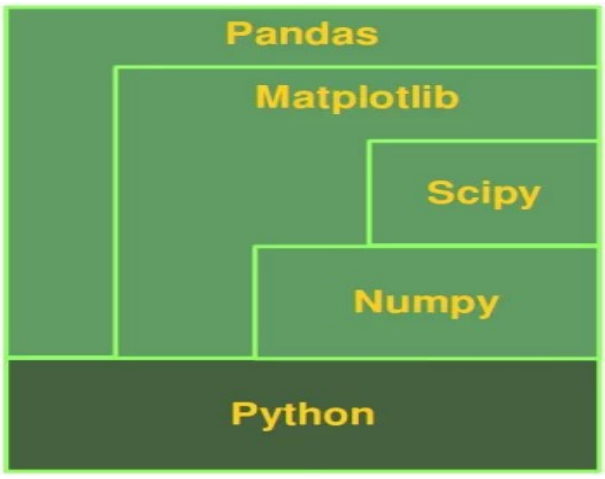

Figura 1: Python y sus módulos numéricos.

Con estas biblitoecas, Python puede ser un excelente reemplazo de Matlab en cuanto a capacidades. La ventaja obvia de Python respecto a Matlab es el costo; Python y estas bibliotecas son software libre y gratuito, Matlab puede alcanzar costos importantes. Pero además, como lenguaje de programación, Python es más moderno y completo, incorpora continuamente nuevos módulos, y se ha establecido como una opción atendible para la programación numérica.

Referencia:

- Klein, Berndt. Numerical programming with Python.

4 Bibliotecas para áreas específicas

Las siguientes referencias destacan algunas bibliotecas de Python para diferentes áreas. Están disponibles en **Pypi** (*Python Package Index*): https://pypi.org/.

Bases de datos.

- **SQLAlchemy**. *The Python SQL Toolkit and Object Relational Mapper*. Un conjunto de herramientas SQL y un mapeo objeto-relacional (*Object Relational Mapper*) con la capacidad y flexibilidad de SQL. Provee un conjunto de patrones para el acceso eficiente a bases de datos.

Grafos y redes.

- **NetworkX**. *Network Analysis in Python*. Permite crear, manipular, y estudiar estructura, funciones y dinámica de redes complejas. Soporta estructuras de datos para grafos, digrafos y multigrafos. Incluye numerosos algoritmos de grafos.

HTML, XML, sitios web.

- **BeatutifulSoup**. Facilita la clasificación (*parsing*) y extracción de datos de documentos HTML y XML. Provee expresiones para iterar, buscar y modificar el árbol de clasificación (*parse tree*).
- **Django**. Permite construir aplicaciones web más rápido y con menos código. Ofrece un conjunto de herramientas de alto nivel para un desarrollo rápido, limpio y efectivo, resolviendo las partes más tediosas del diseño.

- **Requests**. Permite enviar solicitudes (*requests*) HTTP/1.1 en forma sencilla, sin requerir el agregado manual de texto a las URLs ni codificar los datos POST. Mantiene la conexión HTTP activa usando urllib3 (*keep alive and HTTP connection pooling*).
- **Bottle**. Un micro entorno de trabajo web (*web-framework*) simple, liviano y rápido para aplicaciones web pequeñas (WSGI, *Web Server Gateway Interface*, convención de llamadas para servidores web que envían pedidos a aplicaciones web o entornos de trabajo). Bottle ofrece despacho de pedidos con soporte para parámetros URL, plantillas, un servidor HTTP incorporado y adaptadores para muchos sistemas WSGI/servidores-HTTP y máquinas para generación de páginas web desde plantillas (*template engines*).

Inteligencia Artificial, Aprendizaje de Máquina, Redes Neuronales.

- **Keras**. *Deep learning for humans*. Una API diseñada para seres humanos, no máquinas. Sigue las mejores prácticas para reducir la carga cognitiva: APIs simples y consistentes, mínimo número de acciones requeridas por parte del usuario para casos de uso comunes, mensajes de error claros y orientativos. Provee extensa documentación y guías para el desarrollo. Redes neuronales profundas, de uso amigable y estructura modular. Recomendada para aprender sobre redes neuronales profundas.
- **PyTorch**. Provee cálculo tensorial (*tensor computation*) como NumPy, con aceleración de GPU (*Graphics Processing Unit*, unidad de procesamiento gráfico), y redes neuronales profundas con diferenciación automática inversa (*tape-based autograd system, reverse mode automatic differenciation*). Permite reusar paquetes como NumPy, SciPy, y Cython para extender su funcionalidad.
- **scikit-learn**. *Machine learning in Python*. Módulo para aprendizaje de máquina construido sobre SciPy, NumPy y Matplotlib. Provee herramientas simples y eficientes para el análisis de datos predictivo: clasificación, regresión, agrupamiento en conjuntos (*clustering*), reducción dimensional, selección de modelos, preprocesamiento (normalización y extracción de caracteristicas).
- **TensorFlow**. Plataforma extremo a extremo de código abierto para aprendizaje automático. Una biblioteca para cálculo numérico de alta eficiencia. Su arquirectura flexible permite el despliegue sobre variedad de plataformas (CPUs, GPUs, TPUs), desde computadoras de escritorio hasta conjuntos de computadoras (*computer cluster*). Desarrollado originalmente por el grupo Google Brain. Provee una base sólida para el aprendizaje de máquina y el aprendizaje profundo. Su núcleo de cálculo numérico eficiente es usado en muchos áreas científicas.

Procesamiento de datos.

- **Bokeh**. Biblioteca para la visualización interactiva de datos en navegadores web. Permite la construcción de gráficos versátiles, concisos y elegantes con alta eficiencia interactiva en conjuntos grandes o en flujos de datos. Bokeh ayuda a crear gráficos interactivos, resúmenes gráficos (*dashboard*), y aplicaciones de datos.

Procesamiento de imágenes.

- **Pillow**. Agrega capacidad de procesamiento de imágenes al intérprete Python. Basada en PIL (*Python Image Library*), Pillow agrega mejoras y provee un uso más amigable. Soporta muchos formatos de archivo, provee una representación interna eficiente, y una potente capacidad de procesamiento de imágenes. El núcleo está diseñado para el acceso rápido a datos almacenados en formatos básicos de pixel. Constituye una base sólida para una herramienta de procesamiento de imágenes de uso general.
- **opencv-python**. Paquetes de OpenCV para Python, solo para CPU. OpenCV, *Open Source Computer Vision*, es una biblioteca de visión computacional y aprendizaje de máquina. Su objetivo es proveer una infraestructura común para aplicaciones de visión por computadora. Con licencia BSD, permite el uso y modificación a nivel comercial.

- **scikit-image**. Provee un conjunto versátil de rutinas para procesamiento de imágenes en Python. Algunos ejemplos incluyen: imágenes a partir de datos, operaciones sobre arreglos de NumPy, manejo de exposición y canales de color, líneas y bordes, transformaciones geométricas, registro de imágenes, filtros y restauración, detección de rasgos y objetos, segmentación de objetos.

Procesamiento de lenguaje natural.

- **NLTK**. *Natural Language Toolkit*, para construir programas que trabajan con lenguaje humano. Provee APIs de fácil acceso a más de 50 recursos léxicos como WordNet, junto con biliotecas de procesamiento de textos para clasificación, análisis léxico (*tokenization*), reducción a la raíz (*stemming*), etiquetado (*tagging*), análisis sintáctico (*parsing*), y razonamiento semántico (*semantic reasoning*).

Optimización numérica.

- **Aesara**. Biblioteca para definir, optimizar y evaluar eficientemente expresiones matemáticas con arreglos multidimensionales. Ofrece integración estrecha con NumPy, uso transparente de unidad de procesamiento gráfico (*GPU, Graphics Processing Unit*), diferenciación eficiente de funciones, optimización de velocidad y estabilidad, generación dinámica de código C, auto verificación y tests unitarios (*Unit Testing*). Aesara se basa y continúa la biblioteca Theano.

Interfaces gráficas (GUI).

- **wxPython**. Herramientas para desarrollo de interfaces gráficas multiplataforma en Python. Permite crear interfaces gráficas nativas para aplicaciones en Python capaces de correr con mínimas o ninguna modificación en MS Windows, Mac, Linux y otros sistemas tipo Linux.

5 Ejercicios

1. Módulo `os`.

   1) Mostrar directorio actual. Cambiar hacia el directorio `/tmp`. Verificar directorio actual.

   2) Mostrar contenido del directorio `/tmp`.

   3) Crear un directorio `/tmp/nvo_dir`. Verificar que `nvo_dir` está en el contenido de `/tmp`: `"nvo_dir" in os.listdir("/tmp")`.

   4) Crear un archivo `/tmp/nvo_dir/nvo_arch` (no olvidar cerrar el archivo). Mostrar el contenido de `/tmp/nvo_dir`; verificar que `nvo_arch` está en `/tmp/nvo_dir`.

   5) Cambiar de nombre el directorio `/tmp/nvo_dir` por `/tmp/otro_dir`; verificar.

   6) Eliminar el archivo `/tmp/otro_dir/nvo_arch`; verificar.

   7) Eliminar el directorio `/tmp/otro_dir`; verificar.

2. Módulo `sys`.

   1) Escribir un programa `argumentos.py` con el siguiente contenido:

   ```
   import sys
   if __name__ == "__main__":
       print(sys.argv)
   ```

   2) Desde el intérprete de comandos, ejecutar el programa con 0, 1 o varios parámetros.

   3) Verificar que al pasar argumentos numéricos se reciben como caracteres.

   4) ¿Es posible pasar como argumentos los valores `True`, `False` y `None` de Python? ¿Por qué? ¿Cómo haríamos para inicializar variables en esos valores pasando argumentos?