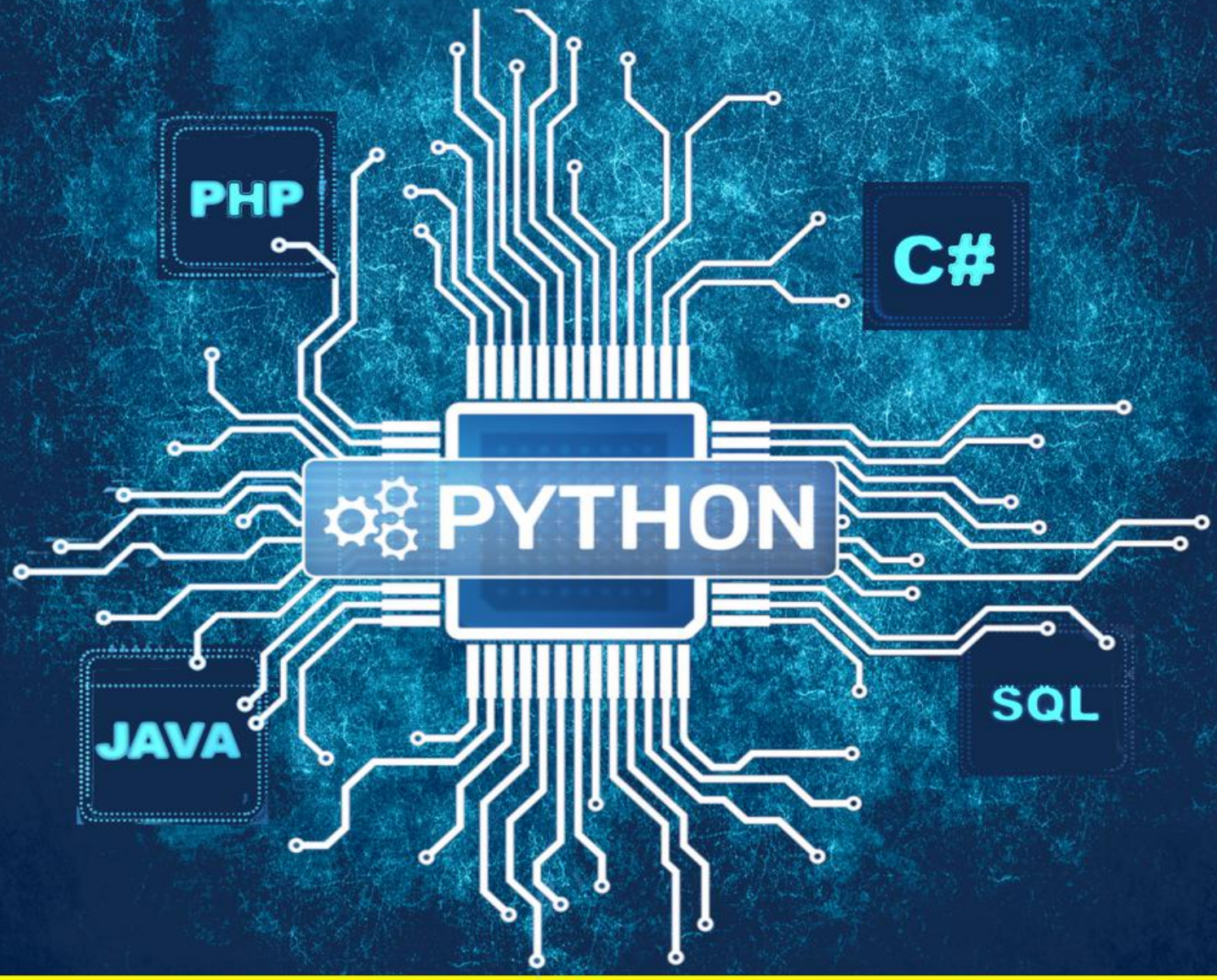
PYTHON

PARA EL ANALISIS DE DATOS

Una Guía para Principiantes para Aprender el Análisis de Datos con la Programación Python.



JOHN HUSH

Python Para el Análisis de Datos

Una Guía para Principiantes para Aprender el Análisis de Datos con la Programación Python.

John Hush

Tabla de Contenido

Tabla de Contenido

Introducción

Felicitaciones por la compra de Python Para el Análisis de Datos y Gracias por hacerlo.

Hay muchos libros sobre este tema en el mercado, ¡gracias de nuevo por elegir este! Se ha hecho todo lo posible para asegurar que esté lleno de tanta información útil como sea posible, ¡por favor disfrútenlo! Python es un lenguaje de alto nivel orientado a objetos (inglés, Lenguaje de Muy Alto Nivel) de mecanografía dinámica, fuerte, interpretada e interactiva.

Características

Python tiene una sintaxis clara y concisa que promueve la legibilidad del código fuente, haciendo el lenguaje más productivo.

El lenguaje incluye una serie de estructuras de alto nivel (listas, diccionarios, fecha/hora, complejos y más) y una gran colección de módulos listos para usar, así como marcos de terceros que pueden ser añadidos. También tiene características que se encuentran en otros idiomas modernos como generadores, introspección, persistencia, metaclase y unidades de prueba. Multiparadigma, el lenguaje soporta programación modular y funcional, así como orientación a objetos. Incluso los tipos básicos en Python son objetos. El lenguaje es interpretado por bytecode por la máquina virtual Python, haciendo el código portátil. Esto hace posible compilar aplicaciones en una plataforma y ejecutarlas en otros sistemas o ejecutarlas directamente desde el código fuente.

Python es un software de código abierto (licenciado bajo la Licencia Pública General (GPL)), pero menos restrictivo, lo que permite que Python se incorpore incluso a productos patentados). La especificación del lenguaje es mantenida por la Fundación de Software Python (PSF).

Además de utilizarse como el principal lenguaje en el desarrollo de sistemas, Python también se utiliza ampliamente como lenguaje de scripts en muchos programas informáticos, lo que permite automatizar tareas y añadir nuevas características, como BrOffice.org, PostgreSQL, Blender, GIMP e Inkscape.

Puedes integrar Python con otros lenguajes, como C y Fortran. En términos generales, el lenguaje tiene muchas similitudes con otros lenguajes dinámicos como Perl y Ruby.

Capítulo 1 – Introducción Análisis de Datos

Para la plataforma Windows, sólo tienes que ejecutar el instalador. Para otras plataformas, como los sistemas Linux, Python suele ser ya parte del sistema, pero en algunos casos, puede ser necesario compilar e instalar el intérprete a partir de los archivos de origen.

También hay implementaciones de Python para.NET (IronPython), JVM (Jython), y Python (PyPy).

Correr Programas

Ejemplo de Programa en Python:

Una lista de instrumentos musicales = [‘Bass’, 'Drums', 'Guitar']

Para cada nombre en la lista de instrumentos:

Mostrar el nombre de los instrumentos musicales e imprimir el instrumento

Salida:

Bajo

Batería

Guitarra

En el ejemplo, "instrumentos" es una lista que contiene los elementos "bajo", "batería" y "guitarra". Ya "instrumento" es un nombre que corresponde a cada uno de los elementos de la lista a medida que se ejecuta el bucle.

Los archivos de origen suelen identificarse con la extensión ".py" y pueden ser ejecutados directamente por el intérprete:

python apl.py

Esto lanzará el programa "apl.py". En Windows, las extensiones de archivo ".py", ".pyw", ".pyc" y ".pyo" se asocian automáticamente con Python durante la instalación, así que sólo hay que hacer clic en el archivo para ejecutarlo. Los archivos ".pyw" se ejecutan con una versión alternativa del intérprete que no abre la ventana de la consola.

Escritura Dinámica

La escritura en Python es fuerte, lo que significa que el intérprete comprueba si las operaciones son válidas y no restringe automáticamente

los tipos incompatibles. Para realizar la operación entre tipos incompatibles, debe convertir explícitamente el tipo de la variable o variables antes de la operación.

Compilación e Interpretación

Por defecto, el intérprete compila el código y almacena el código de bytes en el disco para que la próxima vez que lo ejecute no tenga que volver a compilar el programa, reduciendo el tiempo de carga en la ejecución. Si se cambian los archivos fuente, el intérprete se encargará de regenerar el código de bytes automáticamente, incluso usando el Shell interactivo. Cuando se invoca un programa o módulo, el intérprete realiza el análisis del código, convierte a símbolos, compila (si no hay código de bytes actualizado en el disco), y se ejecuta en la máquina virtual Python.

El código de bytes se almacena en archivos con la extensión ".pyc" (código de bytes normal) o ".pyo" (código de bytes optimizado). El bytecode también puede ser empaquetado junto con el intérprete en un ejecutable para un fácil despliegue de la aplicación, eliminando la necesidad de instalar Python en cada computadora.

Modo Interactivo

El intérprete de Python puede utilizarse de forma interactiva, en el que se escriben líneas de código en un prompt similar al Shell del sistema operativo.

Para evocar el modo interactivo solo hay que ejecutar el intérprete
(si está en camino):
(if it is in path):

Capítulo 2 – Python para el Análisis de Datos

El modo interactivo es una característica distintiva del lenguaje, ya que es posible probar y modificar fragmentos de código antes de incluir el código en los programas, extraer y convertir datos, o incluso analizar el estado de los objetos en la memoria, entre otras posibilidades.

Además del tradicional modo interactivo de Python, existen otros programas que funcionan como alternativas, con interfaces más sofisticadas (como PyCrust):

Herramientas

Existen muchas herramientas de desarrollo para Python, tales como IDEs, editores y Shell (que aprovechan las capacidades interactivas de Python).

Los Entornos de Desarrollo Integrado (IDEs) son paquetes de software que integran varias herramientas de desarrollo en un entorno consistente para aumentar la productividad del desarrollador. Generalmente, los IDEs incluyen características como resaltado de sintaxis (codificado por color según la sintaxis del lenguaje), navegadores de código, Shell integrado y finalización de código (el editor proporciona posibles formas de completar el texto que puede identificar).

Python-supporting IDEs incluyen:

PyScripter

SPE (Stani's Python Editor)

Eric

PyDev (plug-in for the Eclipse IDE).

También hay editores de texto de código de programa especializados, que tienen características como coloración de sintaxis, exportación a otros formatos y conversión de codificación de texto.

Shell es el nombre que se da a los entornos de ejecución de comandos interactivos, que pueden utilizarse para probar pequeñas porciones de código y para actividades como el crujido de datos (extracción de información de interés de masas de datos y posterior traducción a otros formatos).

Además del propio Python Shell estándar, hay otros disponibles:

PyCrust (gráficos).

Ipython (texto).

Los Wrappers son utilidades que se usan para construir ejecutables que incluyen código de bytes, intérprete y otras dependencias, permitiendo que la aplicación se ejecute en máquinas sin Python instalado, lo que facilita la distribución de programas.

Python wrappers incluyen:

Py2exe (Windows only).

cx_Freeze (portatil).

Los Frameworks son colecciones de componentes de software (bibliotecas, utilidades y otros) que están diseñados para ser utilizados por otros sistemas.

Cultura

La comunidad de usuarios de Python ha creado algunas expresiones para referirse a temas relacionados con el lenguaje. En esta jerga, el término Pythonic se utiliza para indicar que algo es compatible con los supuestos de diseño de Python, y Unpythonic significa lo contrario. El usuario del lenguaje es llamado desde el pitonista.

Los objetivos del proyecto fueron resumidos por Tim Peters en un texto llamado Zen de Python, que está disponible en el propio Python a través del comando:

importar esto

El texto hace hincapié en la postura pragmática del Dictador Benévolo por la Vida (BDFL), como se conoce a Guido en la comunidad pitoniana.

Las propuestas de mejora del lenguaje se denominan Propuestas de Mejora de la Pitón (PEP), que también sirven de referencia para las nuevas características que se implementen en el lenguaje.

Además del sitio oficial, otras buenas fuentes de información sobre el idioma son PythonBrasil, el sitio de la comunidad python en el Brasil, con mucha información en portugués, y Python Cookbook, un sitio que almacena "recetas": pequeñas porciones de código para realizar tareas específicas.

Sintaxis

Un programa Python se compone de líneas, que pueden continuar en las siguientes líneas utilizando el carácter de barra invertida (\) al final de la línea o paréntesis, paréntesis o corchetes en las expresiones que utilizan tales caracteres.

El carácter # marca el comienzo del comentario. Cualquier texto después de # será ignorado hasta el final de la línea, excepto los comentarios funcionales.

Los comentarios funcionales se utilizan para:

Cambiar la codificación del archivo fuente del programa añadiendo un comentario con el texto "# - * - codificación": <codificación> - * # -" al principio del archivo, donde <codificación> es la codificación del archivo (normalmente latin1 o utf -8). Se requiere cambiar la codificación para soportar caracteres no ingleses en el código fuente del programa).

Ejemplo de comentarios funcionales:

#! / usr / bin / env python

Una línea de código que muestra el resultado 7 veces 3 imprime 7 * 3

Salida:

21

Ejemplos de líneas discontinuas:

Una línea cortada por una barra invertida = 7 * 3 + \

5 / 2

Una lista (rota por una coma)= ['a', 'b', 'c', 'd', 'e']

Una función rota por comas llamada c = rango (1,11)

imprimir todo en la pantalla

imprimir a, b, c

Salida:

23 ['a', 'b', 'c', 'd', 'e'] [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

El comando de impresión inserta espacios entre las expresiones que se reciben como parámetro y un carácter de nueva línea al final, a menos que reciba una coma al final de la lista de parámetros.

Bloques

En Python, los bloques de código están delimitados por el uso de indentación, que debe ser constante en el bloque de código, pero se considera una buena práctica mantener la coherencia en todo el proyecto y evitar la mezcla de tabulaciones y espacios.

La línea que precede al bloque siempre termina con dos puntos (:) y representa una estructura de control del lenguaje o una declaración de una nueva estructura (una función, por ejemplo).

Ejemplo:

Inicio del Programa

Dos Puntos requeridos

Inicio de un bloque
Otro Bloque
Fin de los dos Bloques
Fin del Programa
For i in list 234, 654, 378, 798: for i in [234, 654, 378, 798]:
Si el resto dividido entre 3 es igual a cero: if i % 3 == 0 :
Print...
print i, '/ 3 =', i / 3
Salida:
234/3 = 78
654/3 = 218
378/3 = 126
798/3 = 266
El operador “%” calcula el módulo (residuo de la división)
Objetos

Python es un lenguaje orientado a los objetos, por lo que las estructuras de datos tienen atributos (los datos en sí) y métodos (rutinas asociadas a los datos). Se accede tanto a los atributos como a los métodos mediante el uso de puntos (.).

Para mostrar un atributo:
print object. attribute
Para ejecutar un método:
object. method (argumentos)
Incluso un método sin argumentos necesita paréntesis:
object. method ()

El punto también es utilizado para acceder a las estructuras del módulo que fueron importadas por el programa.

Control de Flujo

Introduzca la temperatura: 23
Normal

Donde: "Introducir temperatura:" es el mensaje que indica que el programa espera la entrada, "23" es la entrada introducida, y "Normal" es la respuesta del programa.

Si el bloque de código consiste en una sola línea, se puede escribir después de los dos puntos:

if temp < 0 : print 'Freezing...'

A partir de la versión 2.5, Python soporta la expresión:

<variable> = <value 1> if <condition> else <value 2>

Where <variable> recibirá <value 1> if <condition> es verdadero y <value 2> el caso contrario.

Lazos

Los ciclos (loops) son estructuras repetidas, utilizadas comúnmente para procesar colecciones de datos, como líneas de un archivo o registros de una base de datos, que deben ser procesados por el mismo bloque de código.

For

Es la estructura repetitiva más comúnmente usada en Python. La declaración acepta no sólo secuencias estáticas sino también secuencias generadas por iteradores. Los iteradores son estructuras que permiten iteraciones, es decir, el acceso a elementos de una colección de elementos de forma secuencial.

Durante la ejecución de un bucle, la referencia apunta a un elemento de la secuencia. En cada iteración, la referencia se actualiza, de modo que el bloque de código debe procesar el elemento correspondiente.

Tipos

Hay varios tipos de datos simples predefinidos en Python, como:

Números (enteros, reales, complejos...), Texto.

También hay tipos que funcionan como colecciones. Los principales son:

Lista,

Tupla,

Diccionario.

Los tipos de Python pueden ser:

Cambiables: permiten cambiar el contenido de las variables.

Inmutable: no permiten que se cambie el contenido de las variables.

En Python, los nombres de las variables son referencias, que pueden ser cambiadas en tiempo de ejecución.

Los tipos y las rutinas más comunes se implementan en forma de builtins, lo que significa que siempre están disponibles en tiempo de ejecución sin necesidad de importar ninguna biblioteca.

Números

Python tiene una serie de operadores definidos para manipular los números, a través de cálculos aritméticos, operaciones lógicas (que

comprueban si una condición es verdadera o falsa), o procesamiento de bits (donde los números se tratan en forma binaria).

Operaciones aritméticas:

Suma (+).

Diferencia (-).

Multiplicación (*).

División (/): Entre dos números enteros funciona igual que la división de números enteros. En otros casos, el resultado es real.

División entera (//): El resultado se trunca al siguiente número entero inferior, incluso cuando se aplica a números reales, pero en este caso, el resultado también será real.

Módulo (%): Devuelve el resto de la división.

Potencia (**): puede utilizarse para calcular la raíz a través de los exponentes fraccionarios (ejemplo: 100 ** 0,5).

Positivo (+).

Negativo (-).

Los símbolos pueden ser usados para mostrar los números en varios formatos.

Ejemplos:

Ceros al principio

print 'Now it's% 02d:% 02d.' % (16, 30)

Real (el número después del punto controla los decimales)

print "Percentage:%.1f %% Exponential:.% 2e ' % (5.333, 0.00314)

Octal and hexadecimal

print 'Decimal:% d, Octal:% o, Hexadecimal:% x' % (10, 10, 10)

Salida:

Ahora son las 4:30 pm.

Porcentaje: 5.3%, Exponencial: 3.14e-03

Decimal: 10, Octal: 12, Hexadecimal: a

A partir de la versión 2.6, se dispone de otra forma de interpolación además del operador "%", el método de la cadena y la función llamada formato ().

Ejemplos:

musicians = [('Page', 'guitarist', 'Led Zeppelin'), ('Fripp', 'guitarist', 'King Crimson')]

Parámetros identificados en orden msg = '{0} is {1} from {2}'

for name, function, band in musicians :

print (msg. format (name, function, band))
# Parámetros Identificados por Nombre
msg = '{greeting}, are {time: 02d}: {minute: 02d}'
print msg. format (greeting = 'Good morning', hour = 7, minute = 30)
Builtin format () function
print 'Pi =', format (3.14159, '.3e')
Salida:
Page is a guitarist for Led Zeppelin
Fripp is a guitarist for King Crimson
Good morning, it's 07:30
Pi = 3.142e + 00

La función de formato () puede utilizarse para dar formato a un solo dato a la vez.

Se pueden obtener rebanadas (trozos) de cadenas colocando paréntesis después de la cadena de índice.

Los índices en Python:

Comienzan en cero.

Cuentan desde el final si son negativos.

Se pueden definir como estiramientos, en la forma [inicio: fin + 1: intervalo]. Si no se fija el inicio, se considerará cero. Si no se establece el final + 1, se considera el tamaño del objeto. El intervalo (entre caracteres), si no se establece, será 1.

Puede invertir las cadenas utilizando un intervalo negativo:

print 'Python' [:: - 1]
Shows: nohtyP

En el módulo de cadenas se han implementado varias funciones para el manejo de texto.

importación del módulo de cadenas de texto importación de cadenas de texto

El alfabeto

a = string. ascii_letters

Rotando el alfabeto un carácter a la izquierda b = a [1 :] + a [0]

La función maketrans () crea una tabla de traducción entre los caracteres de las dos cadenas que recibió como parámetro.

Los caracteres que faltan en las tablas se copiarán en la salida.

tab = string. maketrans (a, b)
# El mensaje...

msg = " 'This text will be translated..
sera muy extrano.
" '

La función translate () utiliza la tabla de traducción creada por maketrans () para traducir una cadena de impresión. translate (msg, tab)

Salida:

Fttf ufyup tfsá usbevAjep..
Wbj gjdbs cfn ftusboip.

El módulo también implementa un tipo llamado Template, que es una plantilla de cadena que puede ser rellenada a través de un diccionario. Los identificadores están iniciados con el signo de dólar ($) y pueden estar rodeados de claves para evitar confusiones.

Ejemplo:

importar el módulo de la cadena importar la cadena
Crear una plantilla de cadena
st = string. Template ('$ warning happened on $ when')
# Llene el modelo con un diccionario
= st. substitute ({ 'warning' : 'Power outage', 'when' : ' April 3, 2002' })

Muestra:

Lack of electricity happened on April 3, 2002 print s

Puedes usar cadenas mutables en Python a través del módulo UserString, que define el tipo de MutableString:

Importar el modulo ing UserString
import UserString = UserString. MutableString ('Python') s [0] = 'p'
print s # show "python"

Las cadenas cambiantes son menos eficientes que las inmutables porque son más complejas (en términos de estructura), lo que resulta en un mayor consumo de recursos (CPU y memoria).

Para utilizar ambos métodos, es necesario pasar por una codificación compatible. Los más utilizados con el idioma portugués son “latin1” and “utf8”.

Listas

Las listas son colecciones heterogéneas de objetos, que pueden ser de cualquier tipo, incluyendo otras listas.

Las listas en Python son cambiantes y pueden ser modificadas en cualquier momento. Las listas pueden ser cortadas como cadenas, pero

como las listas son cambiantes, se pueden hacer asignaciones a los elementos de la lista.

Sintaxis:

list = [a, b,..., z]

Lista Común de Operaciones:

Una nueva lista: Brit Progs from the 70s progs = ['Yes', 'Genesis', 'Pink Floyd', 'ELP']

La función enumerar () devuelve una tupla de dos elementos para cada iteración: un número de secuencia y un elemento de la secuencia correspondiente.

La lista tiene el método pop () que facilita la implementación de filas y columnas:

list = ['A', 'B', 'C']

print 'list:', list

La lista vacía evalúa a la lista de falsos mientras que la lista:

En las filas, el primer ítem es el primero en salir pop (0) elimina y devuelve el primer ítem de impresión 'Exit', list. pop (0), 'missing', len (list)

Mas listas de Ítems

list + = ['D', 'E', 'F']

print 'list:', list

while list:

En las columnas, el primer artículo es el último que sale pop () elimina y devuelve el último artículo impreso 'Out', list. pop (), 'missing', len (list)

Salida:

list: ['A', 'B', 'C']

Left A, 2 left

Left B, 1 left

Left C, 0 left

list: ['D', 'E', 'F']

Left F, 2 left

Left E, 1 left

Left D, 0 left

Las operaciones de clasificación (ordenar) y anulación (revertir) se realizan en la lista; por lo tanto, no se generan nuevas listas.

Uplas

Similar a las listas, pero sin cambios: No se puede añadir, borrar o asignar elementos.

Sintaxis:

tuple = (a, b,..., z)

Los paréntesis son opcionales.

Particularidad: la tupla con un solo elemento, se representa como:

t1 = (1,)

Los elementos de una tupla pueden ser referenciados de la misma manera que los elementos de una lista:

first_element = tuple [0]

Las listas pueden convertirse en tuplas:

tuple = tuple (list)

Y las tuplas pueden ser convertidas en listas:

list = list (tuple)

Aunque la tupla puede contener elementos mutables, estos elementos no pueden asignarse, ya que esto modificaría la referencia al objeto.

Ejemplo (utilizando el modo interactivo):

t = ([1, 2], 4)

t [0]. append (3)

t

([1, 2, 3], 4)

>>> t [0] = [1, 2, 3]

Rastreo (última llamada más reciente):

File "<input>", line 1, in ?

TypeError: El objeto no soporta la asignación de posiciones

>>>

Las tuplas son más eficientes que las listas convencionales porque consumen menos recursos computacionales (memoria), ya que son estructuras más simples, como cuerdas inmutables sobre cuerdas mutables.

Otros tipos de secuencias

Python también proporciona entre los constructores:

conjunto: secuencia mutable unívoca (sin repeticiones) no ordenada.

frozenset: secuencia inalterable no ordenada y sin cambios.

Cuando una lista se convierte en conjunto, las repeticiones se descartan.

Diccionarios

Un diccionario es una lista de asociaciones compuesta por una sola clave y las estructuras correspondientes. Los diccionarios son modificables, como las listas.

La clave debe ser de tipo inmutable, se suelen utilizar cadenas, pero también pueden ser tuplas o tipos numéricos. Los elementos del diccionario pueden ser tanto cambiantes como no cambiantes. El diccionario Python no garantiza la clasificación de las claves.

Sintaxis:

dictionary = { 'a' : a, 'b' : b,..., 'z' : z }

Estructura:

{ 'Fractal' : 'IFS', 'Reed' : 'Green', 'Limits' : (640, 480), (0, 0): (42, 35)}

Ejemplo de Diccionario:

dic = { 'name' : 'Shirley Manson', 'band' : 'Garbage' }

Elementos de acceso:

print dic ['name']

Elementos de Adicion:

dic ['album'] = 'Version 2.0'

Borrar un elemento de un diccionario:

del dic ['album']

Obtener los Ítems, claves y valores:

items = dic. items ()

Claves = DIC. keys ()

Valores = DIC. values ()

Salida:

Yes => ['Close To The Edge', 'Fragile']

ELP => ['Brain Salad Surgery']

Genesis => ['Foxtrot', 'The Nursery Crime']

King Crimson => ['Red', 'Discipline']

Ejemplo de matriz dispersa:

Matriz dispersa implementada con el diccionario

La matriz dispersa es una estructura que sólo almacena los valores que existen en la matriz

dim = 6, 12

mat = {}

Las tuplas no cambian

Cada tupla representa una posición en la matriz [3, 7] = 3

mat [4, 6] = 5 mat [6, 3] = 7 mat [5, 4] = 6 mat [2, 9] = 4 mat [1, 0] = 9

for lin in range (dim [0]):

for col in range (dim [1]):

Método de obtención (clave, valor)

devuelve el valor de la clave en el diccionario o si la clave no existe,

devuelve el segundo argumento

print mat. get ((lin, col), 0),

print

Salida:

000000000000

900000000000

000000000400

000000030000

000000500000

000060000000

Obtener la matriz dispersa:

Array in the form of string array = " '0 0 0 0 0 0 0 0 0 0 0 0 0 900000000000 000000000400 000000030000 000000500000

000060000000 " '

mat = {}

Romper la Matriz en Líneas:

for lin, line in enumerate (array. splitlines ()):

Envolver las líneas en columnas

for col, column in enumerate (row. split ()):

column = int (column)

Colocar la columna en el resultado si no es cero

if column :

mat [lin, col] = column

print mat

Sume uno en las dimensiones porque el conteo comienza en cero. "Tamaño de la matriz completa":', (lin + 1) * (col + 1) print 'Sparse array

size:', len (mat)

Salida:

{(5,4): 6, (3,7): 3, (1,0): 9, (4,6): 5, (2,9): 4}

Tamaño del Arreglo Completo: 72

Tamaño de la Matriz Dispersa: 5

La matriz dispersa es una buena solución de procesamiento para las estructuras en las que la mayoría de los elementos permanecen vacíos, como las hojas de cálculo.

Verdadero, falso y nulo

En Python, el tipo booleano (bool) es una especialización entera (int). True se llama True y es igual a 1, mientras que False se llama False y es igual a cero.

Los siguientes valores se consideran falsos:

False.

Ninguno (null).

0 (cero).

" (cadena vacía)

[] (lista vacía).

() (tupla vacía).

{} (diccionario vacío).

Otras estructuras con tamaño igual a cero.

Todos los demás objetos fuera de esta lista se consideran verdaderos.

El objeto None, que es del tipo NoneType, Python es nulo y es calificado como falso por el intérprete.

Los operadores booleanos

Con operadores lógicos, es posible construir condiciones más complejas para controlar las desviaciones condicionales y los bucles.

Los operadores booleanos en Python son: y, o, no, es y en.

y : Devuelve un valor verdadero si y sólo si recibe dos expresiones que sean verdaderas.

o : Devuelve un valor falso si y sólo si recibe dos expresiones que son falsas.

not : devuelve falso si recibe una expresión verdadera y viceversa.

is : devuelve verdadero si se le dan dos referencias al mismo objeto y falso en caso contrario.

in : devuelve verdadero si recibe un objeto y una lista, y el objeto aparece una o más veces en la lista y falso en caso contrario.

El valor resultante se calcula en la operación y de la siguiente manera: si la primera expresión es verdadera, el resultado es la segunda expresión, si no la primera. Para el operador o , si la primera expresión es falsa, el resultado será la segunda expresión. En caso contrario, será la primera. Para los otros operadores, el retorno será bool (Verdadero o Falso).

Ejemplos:

```
print 0 and 3 # Muestra 0
print 2 and 3 # Muestra 3
print 0 or 3 # Muestra 3
print 2 or 3 # Muestra 2
print not 0 # Muestra True
print not 2 # Muestra False
print 2 in (2, 3) # Muestra True
print 2 is 3 # Muestra False
```

Además de los operadores booleanos, están las funciones all (), que devuelven verdadero cuando todos los elementos son verdaderos en la secuencia utilizada como parámetro, y any (), que devuelve verdadero si cualquier elemento es verdadero.

Funciones

Las funciones son bloques de código identificados por un nombre que pueden recibir parámetros predeterminados.

En Python, las funciones tienen las siguientes propiedades:

Pueden o no devolver objetos.

Aceptan Doc Strings.

Aceptan parámetros opcionales (con valores por defecto). Si no se pasan, el parámetro será igual al predeterminado definido en la función.

Aceptan que los parámetros se pasen con nombre. En este caso, el orden en el que se pasaron los parámetros no importa.

Tiene su propio espacio de nombres (ámbito local), por lo que puede eclipsar las definiciones de ámbito global.

Puede hacer que sus propiedades sean modificadas (generalmente por los decoradores).

Las cadenas de documentos son cadenas asociadas a la estructura de Python. En las funciones, las Doc Strings se colocan dentro del cuerpo de la función, normalmente al principio. El propósito de las Doc Strings es servir como documentación de esa estructura.

Sintaxis:

def func (parameter1, parameter2 = default):
"" "Doc String
"" "
<code block>
valor de retorno

Los parámetros con valores por defecto deben estar después de los no predeterminados.

Comentarios:

Los argumentos por defecto deben venir en último lugar, después de los argumentos por defecto.

El valor por defecto de un parámetro se calcula cuando se define la función.

Los argumentos pasados sin identificadores son recibidos por la función en forma de lista.

Los argumentos pasados con el identificador son recibidos por la función en forma de diccionario.

Los parámetros pasados con identificadores en la llamada de la función deben venir al final de la lista de parámetros.

Ejemplo:

print eval ('12.2 / 3.3 ')

Salida:

Con esto, es posible ensamblar el código que se pasará al intérprete durante la ejecución de un programa. Esta característica debe ser usada con precaución ya que el código ensamblado a partir de las entradas del sistema abre agujeros de seguridad.

Capítulo 3- Conjunto de Datos

PyDOC es la herramienta de documentación de Python. Se puede utilizar para acceder a la documentación de los módulos que acompañan a Python, así como a la documentación de los módulos de terceros.

En Windows, vaya al icono "Module Docs" para la documentación de la biblioteca estándar y "Manuales de Python" para los tutoriales, referencias y otros documentos más extensos.

Para usar PyDOC en Linux

pydoc./modulo.py

Para mostrar la documentación de "modulo.py" en el directorio actual.

En Linux, la documentación de la biblioteca puede ser vista a través del navegador usando el comando:

pydoc -p 8000

En la direccion http: // localhost: 8000 /.

Para correr la version grafica de PyDOC corre:

pydoc -g

PyDOC utiliza el módulo Doc Strings para generar documentación.

Además, también se puede consultar la documentación en el propio intérprete utilizando la función de ayuda ().

Ejemplo:

help (list)

Muestra la documentación de la lista de Python.

Módulos

Para Python, los módulos son archivos fuente que pueden ser importados a un programa. Pueden contener cualquier estructura Python y se ejecutan al ser importados. Se compilan cuando se importan por primera vez y se almacenan en un archivo (con extensión ".pyc" o “. pyo"), tienen su propio espacio de nombres y soportan Doc Strings. Son objetos Singleton (sólo se carga una instancia en la memoria, que está disponible globalmente para el programa).

Los módulos son localizados por el intérprete a través de la lista de carpetas de PYTHONPATH (sys.path), que normalmente incluye primero la carpeta actual. Los módulos se cargan a través de la declaración de

importación. Por lo tanto, cuando se utiliza alguna estructura de módulo, es necesario identificar el módulo. Esto se denomina importación absoluta.

importarlos

print os. name

También se pueden importar módulos relativamente:

Desde el os import name

Imprimir nombre

El carácter "*" puede ser usado para importar todo lo que se define en el módulo:

Desde el os import *

Imprimir nombre

Al evitar problemas como la ofuscación de las variables, la importación absoluta se considera una mejor práctica de programación que la importación relativa.

Ejemplo de módulo:

Calc.py file

Función definida en def media module (list):

return float (sum (list)) / len (list)

Ejemplo del uso del módulo:

Importar el cálculo import calc module

= [23, 54, 31, 77, 12, 34

Llama la function definida en calc print calc. media (l)

Salida:

38.5

El módulo principal de un programa tiene la variable __nombre__ igual a "__main__", así que puedes probar si el módulo es el principal usando:

if __name__ == "__main__" :

Aquí el código sólo será ejecutado

si este es el módulo principal

no cuando es importado por otro programa

Esto hace que sea fácil convertir un programa en un módulo.

Dividir los programas en módulos facilita la reutilización y la búsqueda de fallos en el código.

Name Scope

El alcance de los nombres en Python se mantiene a través de los Namespaces, que son diccionarios que relacionan los nombres de los objetos (referencias) y los objetos mismos.

Normalmente, los nombres se definen en dos diccionarios, que pueden consultarse a través de las funciones locales () y globales (). Estos diccionarios se actualizan dinámicamente en tiempo de ejecución.

Un espacio de nombres es un ámbito de definición de estructuras.

Se trata de variables globales.

Var_3 fue eclipsada como lo fue

(re) definido en el ámbito local.

Estas son variables locales.

Las variables globales pueden quedar eclipsadas por las locales (ya que el ámbito local se consulta antes que el ámbito global). Para evitar esto, debe declarar la variable como global en el ámbito local.

Ejemplo:

Aunque los diccionarios devueltos por locales () y globales () pueden cambiarse directamente, esto debe evitarse ya que puede tener efectos indeseables.

Somali def (list):

"" "

La suma de las listas es recursiva

Ponga el resultado como global

"" "

Suma global

for item in list :

if type (item) is list : # If item type is Somalist list (item)

else :

sum + = item

sum = 0

Somali ([[1, 2], [3, 4, 5], 6])

print sum # 21

El uso de variables globales no se considera una buena práctica de desarrollo, ya que dificulta la comprensión del sistema, por lo que es mejor evitar su uso. Y ofuscar las variables también.

Paquetes

Los paquetes son carpetas que el intérprete identifica por la presencia de un archivo llamado "__init__.py". Los paquetes actúan como colecciones para organizar los módulos de forma jerárquica.

Se pueden importar todos los módulos del paquete utilizando la declaración de importación de nombre_de_paquete *.

El archivo "__init__.py" puede estar vacío o contener el código de inicialización del paquete o establecer una variable llamada __all__, se importará una lista de módulos del paquete cuando se utilice "*". Sin el archivo, Python no identifica la carpeta como un paquete válido.

Biblioteca Estándar

A menudo se dice que Python viene con "baterías incluidas", en referencia a la vasta biblioteca de módulos y paquetes que se distribuyen con el intérprete. Algunos módulos importantes de la biblioteca por defecto:

Matemáticas: matemáticas, cmath, decimal y aleatorio.

Sistema: os, glob, shutils y subproceso.

Hilos: threding.

Persistencia: pickle y cPickle.

XML: xml.dom, xml.sax y elementTree (a partir de la versión 2.5).

Configuración: ConfigParser y optparse.

Hora: hora y fecha.

Otros: sys, logging, traceback, types y timeit.

Matemáticas

Además de los tipos numéricos incorporados del intérprete, en la biblioteca estándar de Python hay varios módulos dedicados a la realización de otros tipos y operaciones matemáticas.

El módulo matemático define funciones logarítmicas, de exponenciación, trigonométricas, hiperbólicas y de conversión angular, entre otras. El módulo de cmath, por otro lado, implementa funciones similares pero hechas para procesar números complejos.

Las fracciones pueden ser inicializadas de varias maneras: como una cadena, como un par entero o como un número real. El módulo también tiene una función llamada gcd (), que calcula el mayor divisor común (MDC) entre dos enteros.

Archivos y I / O

Los archivos en Python están representados por objetos de tipo de archivo 18, que proporcionan métodos para diversas operaciones de archivo. Los archivos pueden abrirse para leer ('r', que es el valor predeterminado), escribir ('w') o añadir ('a'), en modo texto o binario ('b').

En Python:

sys.stdin representa la entrada estándar.

sys.stdout representa la salida estándar.

sys.stderr representa la salida estándar de errores.

La entrada, la salida y el error estándar son tratados por Python como archivos.

La referencia abierta apunta al archivo.

Abre la entrada en modo de lectura y los otros en modo de escritura.

Los objetos de archivo también tienen un método de búsqueda (), que permite ir a cualquier posición del archivo.

En la versión 2.6, está disponible el módulo io, que implementa por separado las operaciones de archivo y las rutinas de manipulación de texto.

Capítulo 4 - Aplicación de Data Analytic hoy en día

Los sistemas operativos modernos almacenan archivos en estructuras jerárquicas llamadas sistemas de archivos.

En el módulo os.path se implementan varias características relacionadas con los sistemas de archivos, como por ejemplo:

os.path.basename () : Devuelve el componente final de una ruta.

os.path.dirname () : Devuelve una ruta sin el componente final.

os.path.exists () : devuelve True si el camino existe o False si no.

os.path.getsize () : Devuelve el tamaño del archivo en bytes.

El glob es otro módulo relacionado con el sistema de archivos:

```
import them. path
import glob
Muestra una lista de nombres de archivos
y sus respectivos tamaños
for file in sorted ( glob. glob ( '*.py' )):
print file, os. path. getsize ( arq )
```

La función glob.glob () devuelve una lista de nombres de archivos que cumplen el criterio pasado como parámetro, similar al comando "ls" disponible en los sistemas UNIX.

Archivos Temporales

El módulo implementa algunas funciones para facilitar la creación de archivos temporales, liberando al desarrollador de algunas preocupaciones, tales como:

Evitar colisiones con los nombres de los archivos que están en uso.

Identificar el área del sistema de archivos apropiada para los temporales (que varía según el sistema operativo).

Exponer la aplicación a riesgos (el área de temporal es utilizada por otros procesos).

Ejemplo:

importarlos

text = 'Test'

Crea un archive temporal temp = os. tmpfile ()

Escribe en un archivo temporal file. write ('Test')

Retorna al inicio de un archivo temporal file. seek (0)

Muestra el contenido de impresión del archive temporal file. read ()

Cerrar el archivo

temp. close ()

Salida:

test

También existe la función tempnam (), que devuelve un nombre de archivo temporal válido, incluyendo una ruta que respeta las convenciones de los sistemas operativos. Sin embargo, corresponde al desarrollador asegurarse de que la rutina se utilice de manera que no comprometa la seguridad de la aplicación.

Expresiones Regulares

Una expresión regular es una forma de identificar patrones en las cuerdas. En Python, el módulo re proporciona un analizador que permite el uso de tales expresiones. Los patrones definidos por caracteres que tienen un significado especial para el analizador.

Bibliotecas de Terceros

Hay muchas bibliotecas escritas de terceros disponibles para Python, consistentes en paquetes o módulos, que implementan muchas características más allá de la biblioteca estándar.

En general, las bibliotecas se distribuyen de las siguientes maneras:

Paquetes de Distutils.

Paquetes para administradores de paquetes de sistemas operativos.

Instaladores

Huevos de Python.

Los paquetes que usan el módulo de distutils, que se distribuye con Python, son muy populares. Los paquetes se distribuyen en archivos comprimidos (normalmente ".tar.gz", ".tar.bz2" o ".zip"). Para instalarlos, hay que descomprimir el archivo, entrar en la carpeta descomprimida y finalmente ejecutar el comando:

python setup.py install

Que el paquete se instalará en la carpeta "site-packages" en Python.

Los administradores de paquetes del sistema operativo a menudo trabajan con sus propios formatos de paquetes, como ".deb" (Debian Linux) o ".rpm" (RedHat Linux). La forma de instalar los paquetes depende del gestor utilizado. La gran ventaja es que el gestor de paquetes se encarga de las dependencias y las actualizaciones.

Los programas instaladores no son más que ejecutables que instalan la biblioteca. Normalmente se utilizan en el entorno de Windows y pueden ser desinstalados por el Panel de Control.

Python Egg es un formato de paquete (con la extensión ".egg") que es administrado por easy_install, una utilidad que forma parte del proyecto setuptools 19.

Ruby Gems se está convirtiendo poco a poco en el estándar de facto para la distribución de librerías Python.

El programa busca la última versión del paquete en el Índice de paquetes Python (PYPI 20), el repositorio de paquetes Python, y también intenta instalar las dependencias que necesite. Los paquetes Python Eggs se pueden instalar mediante el comando:

easy_install package_name

El script easy_install está instalado en la carpeta de scripts de Python.

Dirección: http://pypi.python.org/pypi.

El manejo de excepciones puede tener un bloque más, que se ejecutará cuando no haya ninguna excepción y un bloque finalmente, se ejecutará de todos modos, habiendo sido una excepción. Se pueden definir nuevos tipos de excepción a través de la herencia de la clase de excepción.

A partir de la versión 2.6, está disponible la sentencia with, que puede anular la combinación try / finally en muchas situaciones. Con 'with', podemos definir un objeto que se utilizará durante la ejecución del bloque.

El objeto debe soportar el protocolo de gestión de contexto, lo que significa que debe tener un método __entrar __ (), que se ejecuta al principio del bloque, y otro llamado __salir __ (), que se evoca al final del bloque.

Capítulo 5 – Matemáticas Para el Análisis de Datos

Las funciones generalmente siguen el flujo convencional de procesamiento, devolución de valores y terminación. Los generadores funcionan de manera similar, pero recuerdan el estado de procesamiento entre las llamadas, permaneciendo en la memoria, y devolviendo el siguiente elemento esperado cuando se activa.

Los generadores tienen varias ventajas sobre las funciones convencionales:

Evaluación perezosa: Los generadores sólo se procesan cuando son realmente necesarios, ahorrando así recursos de procesamiento.

Reducen la necesidad de crear listas.

Permiten trabajar con secuencias ilimitadas de elementos.

Los generadores son generalmente evocados a través de un bucle de for. La sintaxis es similar a la de la función tradicional, excepto que la declaración de rendimiento reemplaza al retorno. En cada nueva iteración, yield devuelve el siguiente valor.

Para convertir la salida del generador en una lista:

list = list (generator ())

Por lo tanto, todos los artículos se generarán a la vez.

El generador de rango () puede anular la función de rango () en la mayoría de los casos, y la sintaxis es la misma, con la ventaja de ahorrar memoria.

Clases

En Python:

Casi todo es objeto, incluso los tipos básicos, como los números enteros.

Los tipos y clases están unificados.

Los operadores son en realidad llamados para métodos especiales.

Las clases son abiertas (menos para los tipos incorporados).

Los métodos especiales se identifican por nombres en el patrón __method __ () (dos guiones bajos al principio y al final del nombre) y definen cómo se comportarán los objetos derivados de la clase en situaciones particulares, como la sobrecarga de los operadores.

Para eliminar una referencia a un objeto, utilice el comando del. Si se eliminan todas las referencias, el Garbage Collector borrará el objeto.

Clases

En Python, hay dos tipos de clases, llamadas estilo antiguo y estilo nuevo. Las clases de estilo nuevo se derivan de la clase de objeto y pueden utilizar nuevas características de las clases de Python, como propiedades y metaclases. Las propiedades son atributos en tiempo de ejecución calculados mediante métodos, mientras que las metaclases son clases que generan clases con las que se puede personalizar el comportamiento de las clases. Las clases de estilo antiguo son un legado de versiones anteriores de Python, mantenidas para garantizar la compatibilidad con el código heredado.

Sintaxis:

```
class Class ( supcl1, supcl2 ):
"" "
Esto es una clase
"" "
clsvar = []
def __init__ ( self, args ):
"" "
Inicializador de Clases
"" "
< code block >
def __done__ ( self ):
"" "
Destructor de clase
"" "
< code block >
def method ( self, params ):
"" "
Método de Objeto
"" "
< code block >
@classmethod
def cls_method ( cls, params ):
"" "
Metodo de Clase
```

""" "
< code block >
@staticmethod
def this_method (params):
""" "
Método Estático
""" "
< code block >
obj = Class ()
obj. method ()
Class. cls_method ()
Class. this_method ()

Los métodos de objetos pueden utilizar atributos y otros métodos de objetos. La auto variable, que representa el objeto y también necesita ser pasada explícitamente. El nombre self es una convención, como cls, y puede ser cambiado por cualquier otro nombre, pero se considera una buena práctica mantener el nombre.

Los métodos de clase sólo pueden utilizar atributos y otros métodos de clase. El argumento cls representa la clase en sí misma, debe ser pasado explícitamente como el primer parámetro del método.

Los métodos estáticos son aquellos que no tienen conexión con los atributos del objeto o de la clase. Funcionan como funciones comunes.

En Python no existen variables y métodos privados (a los que sólo se puede acceder desde el propio objeto). En su lugar se utiliza una convención, el uso de un nombre que comienza con un guión bajo (_) debe considerarse parte de la implementación interna del objeto y está sujeto a cambios sin previo aviso.

Herencia Simple

La herencia es un mecanismo que proporciona la orientación a los objetos, con el fin de facilitar la reutilización del código. La idea es que las clases se construyen formando una jerarquía.

La nueva clase puede implementar nuevos métodos y atributos y heredar los métodos y atributos de la clase antigua (que también puede haber heredado de clases anteriores), pero estos métodos y atributos pueden ser anulados en la nueva clase.

La forma común de herencia se denomina herencia simple, en la que la nueva clase se deriva de una sola clase existente, pero se pueden crear

múltiples clases derivadas mediante la creación de una jerarquía de clases.

Para encontrar métodos y atributos, la jerarquía se sigue de abajo hacia arriba, de forma similar a la búsqueda de los espacios de nombres locales y globales.

Reiniciando y/o creando métodos y/o atributos.

En las clases de estilo antiguo, la resolución comienza con la clase más a la izquierda y baja hasta el final de la jerarquía y luego pasa a la rama derecha.

Ya en las nuevas clases de estilo, la resolución se hace desde la izquierda, descendiendo hasta encontrar la clase común entre los caminos dentro de la jerarquía. Cuando se encuentra una clase común, la búsqueda va por el camino correcto. Cuando los caminos se agotan, el algoritmo procede a la clase común y repite el proceso.

En la jerarquía de clases del ejemplo, el MRO para la clase anfibia será:

[<class '__main __. Amphibious'>,
<class '__main __. Car'>,
<class '__main __. Earth'>,
<class '__main __. Boat'>,
<class '__main __. Aquatic'>,
<type 'object'>]

La herencia múltiple es una característica controvertida porque su uso puede hacer que el proyecto sea confuso y oscuro.

Capítulo 6 – La Discusión de los Datos

Es importante señalar que cuando el proceso muere, todos sus hilos terminan.

En la versión 2.6 también está disponible el módulo de multiprocesamiento, que implementa clases para la creación de procesos y la comunicación entre ellos.

Persistencia

La persistencia puede definirse como el mantenimiento del estado de una estructura de datos entre las ejecuciones de una aplicación. La persistencia libera al desarrollador de escribir código explícitamente para almacenar y recuperar estructuras de datos en archivos y ayuda a mantener el enfoque en la lógica de la aplicación.

Serialización

La forma más simple y directa de persistencia se llama serialización 25 y consiste en escribir en el disco un volcado del objeto, que puede ser cargado posteriormente. En Python, la serialización se implementa de muchas maneras, siendo la más común a través del módulo llamado pickle.

Ejemplo de serialización:

El programa intenta recuperar el diccionario de configuración usando el objeto de archivo “setup.pkl”.

Si tiene éxito, imprime el diccionario.

Si falla, crea una configuración predeterminada y la guarda en “setup.pkl”.

Los módulos estándar de la biblioteca incluyen otros módulos de persistencia, como:

cPickle: versión más eficiente de pickle, pero no puede ser subclasificado.

shelve: proporciona una clase de objetos persistentes similares al diccionario.

Hay marcos Python de terceros que ofrecen formas más persistentes de persistencia, como ZODB.

Todas estas formas de persistencia almacenan datos en formas binarias, que no son directamente legibles por los humanos.

Para almacenar datos en forma de texto, hay módulos Python para leer y escribir estructuras de datos en formatos:

JSON 26 (JavaScript Object Notation).

YAML 27 (YAML Ain't a Markup Language).

XML 28 (Extensible Markup Language).

ZODB

La Base de Datos de Objetos Zope (ZODB) es una base de datos orientada a objetos que ofrece una forma casi transparente de persistencia para aplicaciones escritas en Python y está diseñada para tener poco impacto en el código de la aplicación.

ZODB soporta transacciones, versiones de objetos y puede ser conectada a otros backends a través de Zope Enterprise Objects (ZEO), permitiendo incluso la creación de aplicaciones distribuidas a través de múltiples máquinas en red.

ZODB es un componente integral de Zope 29, que es un servidor para aplicaciones desarrolladas en Python, ampliamente utilizado en Sistemas de Gestión de Contenidos (CMS).

Página de formato en: http://www.json.org/.

Página de formato en: http://yaml.org/.

Página de formato en: http://www.w3.org/XML/.

La documentación y los paquetes de instalación de Zope y los productos conexos en http://www.zope.org/.

Componentes de ZODB:

Base de datos : Permite a la aplicación hacer conexiones (interfaces de acceso a objetos).

Transacción : interfaz que permite hacer cambios permanentes.

Persistencia : Proporciona la clase de base Persistente.

Almacenamiento : Administra la representación persistente del disco.

ZEO : Compartir objetos entre diferentes procesos y máquinas.

Ejemplo de uso de ZODB:

from ZODB import FileStorage, DB

import transaction

Item went back to what it was before the transaction print root ['singer'] # Kate Bush

La ZODB tiene algunas limitaciones que deben tenerse en cuenta durante el diseño de la aplicación:

Los objetos deben ser "serializables" para ser almacenados.

Los objetos cambiantes requieren un cuidado especial.

Los objetos "serializables" son aquellos que pueden ser convertidos y recuperados por Pickle. Entre los objetos que no pueden ser procesados por Pickle están los implementados en módulos escritos en C, por ejemplo.

YAML

YAML es un formato de serialización de datos de texto que representa los datos como combinaciones de listas, diccionarios y valores escalares. Su principal característica es que es legible para el ser humano.

El proyecto YAML fue fuertemente influenciado por la sintaxis de Python y otros lenguajes dinámicos. Entre otras estructuras, la Especificación 30 de YAML establece que:

Los bloques están marcados por una hendidura.

Las listas están entre paréntesis o indicadas por guiones.

Las teclas del diccionario van seguidas de dos puntos.

Las listas pueden representarse de la siguiente manera:

Azul

Blanco

Rojo

O:

[azul, blanco, rojo]

Los diccionarios se representan como:

Color : Blanco

Nombre: Bandido

Raza : Labrador

PyYAML 31 es un módulo de rutinas para generar y procesar YAML en Python.

MVC

El modelo-vista-controlador (MVC) es una arquitectura de software que divide la aplicación en tres partes distintas: el modelo de datos de la aplicación, la interfaz de usuario y la lógica de control.

El objetivo es acoplarse libremente entre las tres partes de modo que un cambio en una parte tenga poco (o ningún) impacto en las otras partes.

MVC: Controlador de la vista del modelo

Recupera los datos y el controlador recibe y reacciona a los eventos del usuario.

La creación de la aplicación depende de la definición de tres componentes:

Modelo (model): encapsula los datos de la aplicación y la lógica del dominio.

Visión (vista): recupera los datos del modelo y presenta al usuario.

Controlador (controller): recibe y reacciona a posibles eventos como las interacciones del usuario y solicita cambios en el modelo y la visión.

Aunque la arquitectura no determina formalmente la presencia de un componente de persistencia, se implica que es parte del componente del modelo.

El uso más común del modelo MVC es en las aplicaciones web basadas en bases de datos que implementan las operaciones básicas llamadas CRUD (Create, Read, Update, and Delete).

Existen varios marcos de trabajo para aumentar la productividad de la creación de aplicaciones siguiendo el MVC, con características como:

Scripts que automatizan las tareas de desarrollo más comunes.

Generación automática de código.

Uso de ORM.

Uso de CSS 49 (Hojas de Estilo en Cascada).

Uso de Javascript asíncrono y XML (AJAX).

Plantillas de aplicación.

Uso de insight para recoger información sobre estructuras de datos y generar formularios con campos con características correspondientes.

Varias opciones están preconfiguradas con valores por defecto adecuados para la mayoría de las aplicaciones.

Capítulo 7 - Scipy, Numpy, Panda

En Python, además de las características matemáticas que forman parte de la distribución estándar, el procesamiento numérico se puede hacer a través de NumPy y otros paquetes que fueron construidos a partir de él.

Numpy

NumPy es un paquete que incluye:
Clase de arreglo.
Clase de matriz.
Varias funciones auxiliares.

Arreglos

La clase de matriz implementa una matriz homogénea mutable con un número arbitrario de elementos, similar a la lista común de Python, pero más potente.

NumPy es la base de varios otros proyectos de código abierto, como Matplolib y SciPy, que complementan a Numpy de varias maneras.

SciPy

SciPy es un paquete que expande NumPy con otras características científicas.

Entre los módulos que forman parte del paquete, tenemos:
linalg: funciones de álgebra lineal.
fftpack: Transformación de Fourier.
integrate: funciones de integración.
interpolar: Funciones de interpolación.
optimize: funciones de optimización.
signal: procesamiento de la señal.
special: funciones especiales (Airy, Bessel, etc.).

Matplotlib

Hay varios paquetes de gráficos de terceros disponibles para Python, el más popular de los cuales es Pylab / Matplotlib.

El paquete tiene dos módulos principales:

matplotlib : módulo que proporciona abstracción orientada a objetos a las características del paquete.

pylab : módulo que ofrece una colección de comandos que se asemeja a Matlab, y es más adecuado para el uso interactivo.

Interfaz Grafica

La Interfaz Gráfica de Usuario (GUI) se ha hecho popular en el entorno de los ordenadores de sobremesa debido a su facilidad de uso y su productividad. Ahora hay muchas bibliotecas disponibles para construir aplicaciones GUI como GTK +, QT, TK, y wxWidgets.

Arquitectura

Las interfaces gráficas a menudo utilizan la metáfora del escritorio, un espacio en dos dimensiones, que está ocupado por ventanas rectangulares que representan aplicaciones, propiedades o documentos.

Window

Las ventanas pueden contener varios tipos de controles (objetos utilizados para interactuar con el usuario o presentar información) y contenedores (objetos que sirven como depósitos para colecciones de otros objetos).

La mayoría de las veces, la interfaz gráfica espera a que ocurran eventos y responde en consecuencia. Los eventos pueden ser el resultado de la interacción del usuario, como los clics del ratón y el arrastre o la escritura, o los eventos del sistema, como el reloj de la máquina. La reacción a cualquier evento se define mediante funciones de devolución de llamada (funciones que se pasan como argumentos a otras rutinas).

Controles más utilizados:

Etiqueta (label): rectángulo que muestra el texto.

Caja de texto : área de edición de texto.

Botón (button): área que puede ser activada interactivamente.

Botón de radio (radio button): un tipo especial de botón, con el que se forman grupos donde sólo se puede apretar uno a la vez.

Botón de comprobación (botón de comprobación): un botón que puede ser comprobado y deshecho.

Barras de desplazamiento : Deslizadores horizontales o verticales utilizados para los rangos de valores numéricos.

Perilla (botón de giro): caja de texto con dos botones con flechas junto a ese decremento e incremento del número en la caja.

Barra de estado (barra de estado): barra para mostrar mensajes, generalmente en la parte inferior de la ventana.

Imagen (imagen): área para la visualización de imágenes.

Los controles pueden tener aceleradores (teclas de acceso directo) asociados.

Contenedores de uso común:

Barra de menú (barra de menú): sistema de menús, normalmente en la parte superior de la ventana.

Fijo (fijo): los objetos permanecen fijos en las mismas posiciones.

Tabla (Tabla): una colección de compartimentos para asegurar los objetos, dispuestos en filas y columnas.

Caja horizontal (caja horizontal): Similar a la tabla, pero sólo una línea.

Case vertical (vertical box): Similar to the table, but only one column.

Notebook (laptop): varias áreas que pueden ser vistas una a la vez a través de pestañas cuando se seleccionan, generalmente en la parte superior.

Barra de herramientas: área con botones para el acceso rápido a las funciones clave de la aplicación.

Para tratar los eventos de tiempo, las interfaces gráficas implementan una característica llamada temporizador (timer) que evoca la función de devolución de llamada después de un cierto tiempo establecido.

PyGTK

GTK + 60 (GIMP Toolkit) es una biblioteca de código abierto escrita en lenguaje C. Diseñada originalmente para ser usada por GIMP 61 , es compatible con las plataformas más ricas en características de hoy en día, incluyendo un constructor de interfaces llamado Glade.

El sitio web del proyecto está en: http://www.gtk.org/ . y los binarios de Windows están disponibles en: http://gladewin32.sourceforge.net/ . La versión para desarrolladores instala Glade.

Dirección oficial del proyecto: http://www.gimp.org/ .

Documentación y fuentes en: http://www.gnome.org/ .

Sistemas UNIX. GTK+ puede ser usado en Python a través del paquete PyGTK 63. Los puertos de la biblioteca para Windows se pueden encontrar en :

PyGTK: http://ftp.gnome.org/pub/gnome/binaries/win32/pygtk/ .

PyGObject: http://ftp.gnome.org/pub/gnome/binaries/win32/pygobject/

PyCairo: http://ftp.gnome.org/pub/gnome/binaries/win32/pycairo/

Aunque es posible crear interfaces completamente usando código, es más productivo construir la interfaz en el software apropiado. Glade genera

archivos XML con la extensión ".glade" que pueden ser leídos por los programas que utilizan GTK+, automatizando el proceso de creación de interfaces gráficas.

Hoja de ruta básica para la construcción de una interfaz:

No hay ningún Glade:

Crea una ventana usando cualquiera de las plantillas disponibles en "Niveles superiores".

Crear contenedores para almacenar los controles.

Crear controles.

Crear los manipuladores para las señales requeridas.

Guardar el archivo con la extensión ".glade".

En Python:

Importar los paquetes requeridos.

Usar GTK para interpretar el archivo XML de Glade.

Crear rutinas para ser usadas como funciones de devolución de llamada.

Asocie las rutinas con los manejadores de correspondencia que se crearon en Glade usando el método signal_autoconnect () .

Habilitar el bucle para procesar eventos con gtk.main () .

Ejemplo (reloj):

No hay Glade:

Haga clic en "Ventana" en "Niveles superiores".

En las propiedades de la ventana:

- Cambie el "Nombre" por "principal" en el "General".

La página web de PyGTK es http://www.pygtk.org/ .

Cambie "Redimensionar" por "Sí".

Cambiar "Posición de la ventana" a "Centro".

Cambiar "Visible" por "Sí" en "Común".

Cambiar el manejador a "on_main_destroy" de la señal "destroy" de "GtkObject" en "Señales".

Haz clic en "Caja vertical" en "Contenedores", luego haz clic en el interior de la ventana y elige el número de elementos igual a 3.

Haga clic en "Barra de menú" en "Contenedores", luego haga clic dentro del espacio vacío superior y elimine los elementos "Editar" y "Ver".

Haga clic en "Barra de estado" en "Control y visualización" y luego haga clic dentro del espacio vacío inferior.

Cambie el nombre de la barra de estado a "sts_data" en "General".

Haga clic en "Etiqueta" en "Control y visualización" y luego haga clic dentro del espacio vacío central.

En las propiedades de la etiqueta, cambie "Name" a "lbl_hora" y "Label" a vacío en "General", "Width Request" a "300" y "Height Request" a "150" en "Common".

En el Inspector (lista de árbol con todos los elementos), eliminar:

“Imageenuitem1”.

“Imageenuitem2”.

“Imageenuitem3”.

“Imageenuitem4”.

“Separatormenuitem1”.

En el Inspector:

encontrar "imageenuitem5" y cambiar el manejador en "Señales" de la señal "activar" a "on_imagemenuitem5_activate" de "GtkMenuItem".

encontrar "imageenuitem10" y cambiar el handler en "Señales" de la señal "activate" a "on_imagemenuitem10_activate" de "GtkMenuItem".

Guarda el archivo como "clock.glade".

wxPython

El paquete wxPython es básicamente un envoltorio para el conjunto de herramientas (conjunto de herramientas y librerías) wxWidgets, desarrollado en C ++. Características principales:

Multiplataforma.

Aspecto nativo, es decir, consistente con el entorno en el que se ejecuta.

Una gran colección de componentes listos para usar.

Una comunidad muy activa.

La forma general de funcionamiento es similar a GTK +: el marco controla la interacción del usuario a través de un bucle que detecta los eventos y activa las rutinas correspondientes.

La forma más común de implementar una interfaz gráfica a través de wxPython es:

Importar el paquete wx .

Crear una clase de ventana a través de la herencia. En la inicialización de la clase se pueden definir los controles y contenedores que forman parte de la ventana y las asociaciones entre eventos con las funciones de devolución de llamada correspondientes , que suelen ser métodos de la propia clase .

Crear un objeto "aplicación" utilizando la clase wxPython App .

Crear un objeto de la clase window .

Iniciar el bucle de manejo de eventos de la aplicación .

Otras funcionalidades asociadas a la interfaz gráfica se pueden obtener utilizando otros módulos, como pySystray 69 , que implementa una funcionalidad que permite a la aplicación utilizar la bandeja del sistema en Windows.

CG

Gráficos por Computadora (CG) es el área de la Ciencia de la Computación que estudia la generación, representación y manipulación de contenido visual en sistemas de computación y tiene aplicaciones en varias áreas del conocimiento humano.

Las simulaciones, por ejemplo, son sistemas que emplean cálculos matemáticos para imitar uno o más aspectos de un fenómeno o proceso que existe en el mundo real. Las simulaciones permiten comprender mejor cómo funciona el experimento real y comprobar escenarios alternativos con otras condiciones.

En el caso de los juegos, que son en realidad una forma de simulación interactiva que utiliza medios visuales para aumentar la sensación de realismo, lo que se conoce como inmersión, y por lo tanto enriquece la experiencia del jugador.

Otra aplicación es la visualización, como dice un viejo dicho popular: "una imagen vale más que mil palabras", y esto es aún más cierto cuando se trata de interpretar grandes cantidades de datos, como es el caso de muchas actividades científicas, médicas y de ingeniería.

Áreas como la geografía, la cartografía y la geología requieren Sistemas de Información Geográfica (SIG), que representan topologías y datos asociados como la altura, la humedad y otros.

La ingeniería y las actividades conexas utilizan herramientas de diseño asistido por computadora (CAD) para facilitar la creación de dibujos técnicos de componentes o piezas de maquinaria.

Además, varias formas de arte se benefician de la CG, como el cine, especialmente para la creación de efectos especiales. El CG también ha permitido el surgimiento de nuevas formas de arte que utilizan un entorno digital como medio, como la animación 3D.

Arreglos vs Vectores

Es muy común representar la información visual de forma bidimensional (2D), ya sea en fotos, gráficos impresos o en una pantalla LCD. Existen dos formas de representación de imágenes bidimensionales ampliamente utilizadas, cada una con sus ventajas y desventajas.

La primera matriz también se conoce como bits de mapa (bitmap) o raster , en la que la imagen se representa como una matriz bidimensional de puntos con información de color, llamados elementos de imagen (picture element , comúnmente abreviado como pixel). Esta forma requiere la manipulación y el almacenamiento de sofisticados algoritmos, debido al volumen de los datos y a la complejidad de las operaciones, como la interpolación de valores durante el cambio de tamaño, por ejemplo.

La segunda forma de representación es la de las imágenes vectoriales, que se describen a través de entidades matemáticas que conforman la geometría de la imagen (líneas, polígonos, texto y otros). Esta forma es menos exigente en cuanto a recursos computacionales y no presenta problemas asociados con el escalamiento, pero no permite muchas operaciones que el mapa de bits permite.

Entre otras formas de representación, es interesante destacar los fractales, en los que las imágenes se generan a través de algoritmos que se aplican de forma similar.

Estas formas de representación han dado lugar a la aparición de diversos formatos de archivo para almacenar imágenes, incluso abiertas, como Portable Network Graphics (PNG), que admite incluso imágenes raster transparentes , y Scalable Vectorial Graphics (SVG) para imágenes vectoriales. , mapas de bits, e incluso animaciones. Ambos están aprobados por el Consorcio de la World Wide Web (W3C).

Hay ahora varias bibliotecas orientadas a CG disponibles para Python, que están en una etapa avanzada de madurez.

Procesamiento de Imágenes

La Biblioteca de Imágenes de Pitón (PIL) es una biblioteca de procesamiento de matrices para Pitón.

PIL tiene módulos que implementan:

Herramientas para recortar, redimensionar y fusionar imágenes.

Algoritmos de conversión, que soportan varios formatos.

Filtros, como suavizar, desenfocar y detectar bordes.

Ajustes, incluyendo brillo y contraste.

Operaciones con paletas de color.

Dibujos 2D simples.

Rutinas para el procesamiento de imágenes: ecualización, autocontraste, deformación, inversión y otras.

Ejemplo de procesamiento de imágenes:

- * - coding: latin-1 - * -

"" "

Crea miniaturas suavizadas para cada JPEG de la carpeta actual.

"" "

import glob

Importación de la PIL Módulo principal de la imagen

Import ImageFilter filter module

SVG

Los archivos SVG pueden almacenar varios tipos de información vectorial, incluidos los polígonos básicos, que están representados por líneas que delimitan un área cerrada, como rectángulos, elipses y otras formas simples. Además, también admite caminos (paths), que son figuras con relleno o no, compuestas por líneas y/o curvas definidas por puntos, que se codifican mediante comandos de un carácter ("L" significa "Line To" por ejemplo) y un par de coordenadas X e Y, lo que genera un código muy compacto.

El texto Unicode puede incluirse en un archivo SVG, con efectos visuales, y la especificación incluye el manejo de texto bidireccional, vertical y siguiendo trayectorias curvas. El texto se puede formatear con fuentes externas de texto, pero para aliviar el problema de que el texto no se muestre correctamente en los diferentes sistemas, hay una fuente interna que siempre está disponible.

Las figuras geométricas, los trayectos y el texto pueden utilizarse como contornos, por dentro o por fuera, que pueden utilizar tres tipos de relleno:

Colores sólidos, que pueden ser opacos o transparentes.

Gradientes, que pueden ser lineales o radiales.

Patrones, que son imágenes de mapa de bits o vectoriales que se repiten en todo el objeto.

Se pueden animar tantos gradientes como patrones.

SVG también permite al autor incluir metadatos con información sobre la imagen, como el título, la descripción y otros, para facilitar la catalogación, la indexación y la recuperación de archivos.

Todos los componentes de un archivo SVG pueden ser leídos y cambiados usando scripts de la misma manera que el HTML, por defecto en el lenguaje ECMAScript . La especificación también proporciona el manejo de eventos de ratón y teclado , que, junto con los hipervínculos, permite añadir interactividad a los gráficos.

El formato también soporta la animación a través de ECMAScript , que puede transformar los elementos de la imagen y cronometrar el movimiento. Esto también se puede hacer a través de los recursos propios de SVG utilizando etiquetas .

Para el SVG, los filtros son conjuntos de operaciones gráficas que se aplican a un gráfico vectorial determinado para producir una imagen matriz con el resultado. Estas operaciones gráficas se denominan primitivas de filtro, que suelen realizar una forma de procesamiento de la imagen, como el efecto de Desenfoque Gaussiano , y así generan un mapa de bits con transparencia (estándar RGBA) como salida, que se regenera si es necesario. El resultado de una primitiva puede utilizarse como entrada de otra primitiva, permitiendo la concatenación de varias para generar el efecto deseado.

SVGFig

Los archivos SVG pueden manipularse a través de bibliotecas XML, como ElementTree, pero es más productivo utilizar componentes ya diseñados para este fin. SVGFig es un módulo SVG listo para ser utilizado. El módulo permite utilizar directamente tanto primitivas de dibujo SVG como rutinas propietarias de alto nivel.

SVGFig tiene varias primitivas de dibujo implementadas en forma de funciones, entre las que se incluyen rutas (Path ()), líneas (Line ()) y texto (Text ()).

Imágenes Tridimensionales

Los formatos matriciales y vectoriales representan imágenes bidimensionales en la computadora que son adecuadas para la mayoría de las aplicaciones. Sin embargo, están limitados en muchos aspectos, especialmente para las simulaciones, ya que el mundo en el que vivimos tiene tres dimensiones (3D).

Una escena 3D está compuesta por objetos que representan sólidos, fuentes de luz y cámaras. Los objetos sólidos suelen representarse con mallas, que son un conjunto de puntos (vértices). Estos tienen coordenadas x, y, y z. Los puntos están interconectados por líneas (bordes) que forman las superficies (caras) de los objetos. Los conjuntos de líneas que representan las mallas se denominan "wireframes".

Objetos

Los objetos pueden utilizar uno o más materiales, y éstos pueden tener varias características, como el color, la transparencia y el sombreado, que es la forma en que el material responde a la iluminación de la escena. Además, el material puede tener una o más texturas asociadas.

Las texturas están compuestas por imágenes bidimensionales que pueden ser utilizadas en materiales aplicados a las superficies de los objetos, cambiando varias propiedades como el reflejo de la superficie, la transparencia y la protuberancia.

En una escena tridimensional, los objetos pueden modificarse mediante transformaciones como la traslación (pasar de una posición a otra), la rotación (girar alrededor de un eje) y el cambio de tamaño (redimensionar una o más dimensiones).

Wireframe Con imagenes solidas

Para renderizar, es decir, generar la imagen final. Necesitas hacer una serie de cálculos complejos para aplicar la iluminación y la perspectiva a los objetos de la escena. Entre los algoritmos utilizados para la renderización, uno de los más conocidos es el llamado raytrace , en el que se calculan los rayos de luz desde la cámara hasta las fuentes de luz. Esto evita cálculos innecesarios de rayos que no llegan a la cámara.

Uno de los usos más populares de la tecnología 3D es en las animaciones. La técnica de animación 3D más común se llama "keyframe". En él, el objeto a animar se posiciona en diferentes lugares en los momentos clave de la animación, y el software se encarga de calcular los cuadros intermedios.

Muchas aplicaciones 3D utilizan bibliotecas que implementan la especificación OpenGL (Open Graphics Library), que define una API independiente de la plataforma y el lenguaje para manipular los gráficos

3D, lo que permite un renderizado en tiempo real acelerado por hardware. Su característica más llamativa es el rendimiento.

VPython

VPython es un paquete que permite crear y animar modelos tridimensionales simples. Su objetivo es facilitar la rápida creación de simulaciones y prototipos que no requieren soluciones complejas.

VPython proporciona iluminación, control de cámara y manejo de eventos de ratón (rotación y zoom) de forma automática. Los objetos pueden ser creados de forma interactiva en el intérprete, que la ventana 3D de VPython se actualiza en consecuencia.

VPython tiene varias limitaciones. No proporciona formas de crear y/o manipular materiales o texturas más complejas, ni formas avanzadas de iluminación o detección de colisiones. Para escenas más sofisticadas, hay otras soluciones, como Python Ogre 75 y Blender, que es una aplicación de modelado 3D que utiliza Python como lenguaje de scripting .

Py OpenGL

Las bibliotecas OpenGL 76 implementan una API de manipulación de imágenes 3D de bajo nivel, que permite el acceso a los recursos disponibles en el hardware de vídeo , y también hacen que el código sea independiente de la plataforma, ya que el software emula características no disponibles en el equipo. Estas características incluyen primitivas (líneas y polígonos), mapeo de texturas, operaciones de transformación e iluminación.

OpenGL funciona en un contexto, cuyo estado se modifica a través de las funciones definidas en la especificación. Este estado se mantiene hasta que se vuelve a cambiar.

Además de la biblioteca principal, la OpenGL Utility Library (GLU) es una biblioteca de alto nivel, mientras que el OpenGL Utility Toolkit (GLUT) define rutinas independientes de la plataforma para la gestión de ventanas, entradas y contextos.

La GLUT está orientada a eventos, a los que se pueden asociar funciones de devolución de llamada , que ejecutan llamadas OpenGL. La biblioteca tiene una rutina que monitoriza los eventos y evoca las funciones cuando es necesario.

Py OpenGL es un paquete que permite a los programas Python utilizar las bibliotecas OpenGL, GLU y GLUT.

Borra la matriz de proyección glLoadIdentity ()

Calcula el aspecto de la perspectiva gluPerspective (45. , flotador (x) / flotador (y), 0.1 , 100.0)

Selecciona la matriz de visualización glMatrixMode (GL_MODELVIEW)

```
def draw ():
"" "
Funcion que Dibuja Objetos
"" "
Global air , dr
# Ventana despejada y amortiguador de profundidad
glClear ( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT )
Clears the glLoadIdentity () view matrix
Mover el objeto (traducción)
Parámetros: x, y and z (cambio) glTranslatef (- 0.5 , - 0.5 , - 4. )
Rotación (en grados)
Parámetros: grados, x, y and z (ejes) glRotatef ( ar , 1.0 , 1.0 , 1.0 )
Escalas
Parámetros: x, y and z (tamaño)
glScalef ( air / 1000 , air / 1000 , air / 1000 )
for i in xrange ( 0 , 360 , 10 ):
Rotation of glRotatef Object Faces ( 10 , 1.0 , 1.0 , 1.0 )
Starts creating a glBegin rectangular face ( GL_QUADS )
```

Defines the color to use to draw (R, G, B) glColor3f (.7 , .5 , .1)
Create a face vertex
glVertex3f (0. , 0. , 0.)
glColor3f (.7 , .3 , .1)
glVertex3f (1. , 0. , 0.)
glColor3f (.5 , .1 , .1)
glVertex3f (1. , 1. , 0.)
glColor3f (.7 , .3 , .1)
glVertex3f (0. , 1. , 0.)
Ends the face glEnd ()
Inverts the variation if ar > 1000 : dr = - 1 if ar < 1 : dr = 1
air = air + dr
Intercambiar el buffer, mostrando lo que se acaba de usar glutSwapBuffers ()
def keyboard (* args):
"" "
Función de llamada para manejar los eventos del teclado
"" "
Prueba si la tecla ESC fue presionada if args [0] == '\ 33' :
raise SystemExit
if __name__ == '__main__' :
Inicializa GLUT glutInit (argv)
Seleccionar el modo de display
glutInitDisplayMode (GLUT_RGBA | GLUT_DOUBLE | GLUT_DEPTH)
Establece la resolución de la ventana de GLUT de 640 x 480 glutInitWindowSize (640 , 480)
Crear una GLUT window
window = glutCreateWindow ('Transformations')
# Configurar la función de llamada que se basa en la ventana actual

glutDisplayFunc (draw)

Para ver la pantalla completa #glutFullScreen ()
Registra la función para manejar el redibujo de la ventana cuando no hay nada que hacer
glutIdleFunc (draw)
Registra la función para redibujar la ventana cuando se redimensiona
glutReshapeFunc (resize)
Función de registro para manejar los eventos del teclado glutKeyboardFunc (keyboard)
Inicialización de Ventana
Limpia la imagen (black background)
glClearColor (0. , 0. , 0. , 0.)
Limpia el buffer profundo glClearDepth (1.)
Configura el tipo de analisis profundo glDepthFunc (GL_LESS)
Permite el análisis profundo
glEnable (GL_DEPTH_TEST)
configura el sombreado suave glShadeModel (GL_SMOOTH)
Selecciona la matriz de proyección glMatrixMode (GL_PROJECTION) glLoadIdentity ()
gluPerspective (45 , 640 / 480 , .1 , 100.)
Selecciona la matriz de visualización glMatrixMode (GL_MODELVIEW)
Inicie el evento GLUT glutMainLoop () ()
Salir de la ventana:

Imágenes Tridimensionales 267

OpenGL puede integrarse con conjuntos de herramientas gráficas , como wxWidgets y Qt , en lugar de usar GLUT, y así incorporarse a las aplicaciones GUI convencionales.

Una de las principales referencias sobre OpenGL es el libro "Guía de Programación de OpenGL", también conocido como "Libro Rojo" 78 .

La versión online está disponible en http://www.glprogramming.com/red/ .

Procesamiento distribuido

Generalmente la solución a los problemas que requieren mucha potencia de cálculo es el uso de máquinas más potentes, pero esta solución es limitada en cuanto a la escalabilidad. Una alternativa es dividir los procesos de aplicación entre múltiples máquinas que se comunican a través de una red, formando un clúster o una red.

La diferencia básica entre el cluster y la grid es que el primero se basa en la premisa de que el diseño debe ser un entorno controlado, homogéneo y predecible, mientras que el segundo es generalmente heterogéneo, incontrolado e impredecible. Un cluster

un entorno diseñado específicamente para el procesamiento distribuido, con máquinas dedicadas en un solo lugar. Una red se caracteriza por el uso de estaciones de trabajo que pueden estar en cualquier lugar.

El modelo computacional tiene por objeto utilizar los procesadores y la memoria del equipo involucrado para obtener más potencia de cálculo. La aplicación suele utilizar un sistema de metaplanificador, que programa las tareas que han de ser procesadas por los nodos (máquinas que componen el modelo), por lo que la operación tiende a ser continua, con una interacción reducida con los usuarios. Un ejemplo conocido es SETI @ home 79 .

El recurso de clúster se utiliza para almacenar información en un grupo de computadoras, tanto para aumentar el rendimiento de la recuperación de datos como para ampliar la capacidad de almacenamiento. Esta plantilla puede utilizarse para proporcionar una infraestructura para las aplicaciones o para responder a las solicitudes formuladas de manera interactiva por los usuarios. Entre los servicios que pueden funcionar de esta manera se encuentran los sistemas de gestión de bases de datos (DBMS), como el MySQL Cluster 80 .

El modelo híbrido es una aplicación diseñada específicamente para trabajar con

Página del proyecto en: http://setiathome.berkeley.edu/ .

Dirección de Internet: http://www.mysql.com/products/database/cluster/ .

en múltiples máquinas a la vez. En lugar de proporcionar recursos directamente, la aplicación utiliza el equipo para apoyar su propia funcionalidad. Como resultado, la infraestructura es utilizada de manera casi transparente por los usuarios que utilizan la aplicación de manera interactiva. Todos los nodos ejecutan la aplicación y pueden funcionar como servidores y clientes. El ejemplo más común de arquitectura híbrida son los sistemas de intercambio de archivos que utilizan la comunicación entre pares (P2P).

Independientemente del modelo utilizado. Los sistemas distribuidos deben cumplir cuatro requisitos básicos:

Comunicación: Las máquinas involucradas deben comunicarse de tal manera que permitan el intercambio de información entre ellas.

Metadatos: El procesamiento de los datos debe mantenerse adecuadamente.

Control: Los procesos deben ser administrados y controlados.

Seguridad: La confidencialidad, la integridad y la disponibilidad deben ser protegidas.

Existen varias tecnologías enfocadas al desarrollo de aplicaciones distribuidas, tales como: XML-RPC 81 , Web Services, objetos distribuidos, MPI, y otros.

Especificaciones en http://www.xmlrpc.com/ .

Objetos distribuidos

La premisa básica de la tecnología de objetos distribuidos es hacer que los objetos estén disponibles para que sus métodos puedan ser evocados remotamente desde otras máquinas o incluso otros procesos en la misma máquina utilizando para ello la pila de protocolos de red TCP / IP.

Existen varias soluciones para estos casos, pero el uso de objetos distribuidos ofrece varias ventajas sobre otras soluciones que implementan una funcionalidad similar, como el protocolo XML-RPC:

Simplicidad de implementación.

Oculta las capas de comunicación.

Soporte de estructuras de datos nativas (siempre que sean serializables).

Buen rendimiento.

Madurez de la solución.

PYthon Remote Objects (PYRO 82) es un marco de trabajo para aplicaciones distribuidas.

82 Documentación y fuentes disponibles en: http://pyro.sourceforge.net/ .

que le permite publicar objetos a través de TCP / IP. En la máquina del servidor, PYRO publica el objeto, cuidando detalles como el protocolo, el control de sesión, la autenticación, el control de concurrencia y otros.

La comprensión de la lista es más eficiente que el empate tradicional.

Otra forma de mejorar el rendimiento de una aplicación es usando Psyco , que es una especie de Just In Compiler (JIT). En tiempo de ejecución, trata de optimizar el código de la aplicación, por lo que el módulo debe ser importado antes que el código a optimizar (el comienzo del módulo principal de la aplicación es un

85 El módulo cProfile no es adecuado para evaluar fragmentos cortos de código. El tiempo que modula es el más adecuado porque ejecuta el código varias veces durante la evaluación.

lugar apropiado).

Ejemplo (con el último fragmento de código evaluado en el ejemplo anterior):

importar psyco

Trata de optimizar todo lo que sea psicológico.

importar timeit

Lista de cuadrados del 1 al 1000

cod = '[x ** 2 for x in xrange (1, 1001)]' print timeit . Timer (cod). timeit ()

Salida:

127.678481102

El código corrió más del doble de rápido que antes. Esto requirió sólo añadir dos líneas de código.

Sin embargo, Psyco debe ser usado con cierta precaución, ya que en algunos casos, puede no ser capaz de optimizar o incluso empeorar el rendimiento 86 . Las funciones de mapa () y filtro () deben ser evitadas, y los módulos escritos en C, como re (expresiones regulares), deben ser marcados con la función can not compile () para que Psyco los ignore. El módulo proporciona formas de optimizar sólo ciertas partes del código de la aplicación, como la función de perfil () , que sólo optimiza las partes más pesadas de la aplicación y una función de registro () que analiza la aplicación para evitar estas situaciones.

Algunos consejos de optimización:

Mantenga el código simple.

Sólo optimice el código cuando el rendimiento de la aplicación sea realmente crítico.

Hay algunas funciones en las que Psyco tiene el efecto de ralentizar el programa porque el propio Psyco también consume CPU. Además, Psyco también hace que la aplicación consuma más memoria.

Usar herramientas para identificar los cuellos de botella en el código.

Evitar las funciones recursivas.

Utilizar funciones en el idioma nativo. Las listas y diccionarios en Python están muy optimizados.

Utilice Comprensiones de listas en lugar de bucles para procesar listas utilizando expresiones simples.

Evite las funciones dentro de los bucles. Las funciones pueden recibir y devolver listas.

Use generadores en lugar de funciones para grandes cadenas de datos.

Empaquetado y distribución

Suele ser mucho más sencillo distribuir aplicaciones en forma binaria, donde simplemente se ejecuta un ejecutable para lanzar la aplicación que instalar todas las dependencias necesarias en cada máquina en la que se desea ejecutar la aplicación.

Existen programas que permiten generar ejecutables a partir de un programa hecho en Python, como Py2exe 87 y cx_Freeze 88 .

Py2exe funciona sólo en la plataforma Windows, pero tiene muchas características y puede generar ejecutables con una interfaz de texto, gráficos, servicios (programas que se ejecutan sin la intervención del usuario, de manera similar a los demonios de los sistemas UNIX) y servidores COM (arquitectura de componentes de Microsoft).

87 La documentación, las fuentes y los binarios para la instalación se pueden encontrar en

http://www.py2exe.org/ .

88 La documentación de instalación multiplataforma, las fuentes y los binarios se pueden encontrar en:
http://starship.python.net/crew/atuining/cx_Freeze/ .

284 Empaquetado y distribución

Cx_Freeze es portátil y puede funcionar en entornos UNIX, pero es mucho menos versátil que Py2exe.

Para usar Py2exe, es necesario crear un script , comúnmente llamado "setup.py", que le dice a Py2exe lo que se necesita para generar el ejecutable.

Ejemplo de “setup.py”:

```
- * - coding: latin1 - * -
    "" "
    Py2exe usage example
    "" "
    from distutils . core import setup import py2exe
    setup ( name = 'SIM - Interactive Music System' , service = [
'simservice' ],
    console = [ 'sim.py' , 'simimport.py' ],
    windows = [ 'simgtk.py' ],
    options = { 'py2exe' : {
    'optimize' : 2 ,
    'includes' : [ 'atk' , 'gobject' , 'gtk' , 'gtk.glade' , 'pango' , 'cairo' ,
'pangocairo' ]
    }},
    data_files = [( '' , [ 'Janela.glade' , 'sim.ico' ])], description = 'First
Version ...' , version = '1.0' )
```

En el ejemplo, tenemos un sistema que consiste en dos utilidades de línea de comando, una aplicación de interfaz gráfica y un servicio. La aplicación de interfaz gráfica depende de GTK + para funcionar y fue desarrollada usando Glade.

Entre los parámetros de Py2exe, los más comunes son:

nombre : nombre de la aplicación.

service : lista de servicios.

console : lista de programas con una interfaz de texto.

windows : lista de programas con una interfaz gráfica.

options ['py2exe'] : Diccionario con opciones que cambian el comportamiento de Py2exe:

- optimize : 0 (optimización desactivada, bytecode por defecto), 1 (optimización

habilitado, equivale al parámetro "-O" del intérprete) o 2 (optimización con la eliminación de Doc Strings habilitada, equivale al parámetro "-OO" del intérprete).

incluye : lista de módulos que se incluirán como dependencias. Py2exe suele detectar las dependencias sin tener que utilizar esta opción.

archivos_de_datos : Otros archivos que forman parte de la aplicación, como imágenes, iconos y archivos de configuración.

description : comentario.

version : una versión de la aplicación como cadena .

Para generar el ejecutable, el comando es:

```
python setup.py py2exe
```

Py2exe creará dos carpetas:

build : archivos temporales.

dist : archivos para distribución.

Entre los archivos para la distribución, "w9xpopen.exe" sólo es necesario para las versiones antiguas de Windows (95 y 98) y puede ser

eliminado sin problemas en las versiones más recientes.

A través de la línea de comandos, también se pueden pasar algunas opciones interesantes, como el parámetro "-b1", para generar menos archivos para su distribución.

Cx_Freeze es una utilidad de línea de comandos.

FreezePython -OO -c sim.py

La opción "-c" hace que el código de bytes se comprima en formato zip .

FreezePython -OO—include-modules = atk, cairo, pango, pangocairo simgtk.py

La opción "—incluir-módulos" permite pasar una lista de módulos que serán incluidos en la distribución.

Tanto Py2exe como cx_Freeze no son compiladores. Lo que hacen es empaquetar los bytecodes de la aplicación, sus dependencias y el intérprete en (al menos) un archivo ejecutable (y archivos de ayuda) que no dependen del entorno en el que se generaron. Esto hace que la distribución de la aplicación sea mucho más sencilla. Sin embargo, no hay ganancia de rendimiento en la generación de los ejecutables, excepto por la carga de la aplicación a la memoria en algunos casos.

Tampoco generan programas de instalación. Esto requiere el uso de un software específico. Los instaladores son generados por aplicaciones que automatizan tareas comunes en el proceso de instalación. Ejemplos de software en esta categoría son Inno Setup 89 y NSIS 90 .

89 documentación y binarios de instalación disponibles en:
http://www.jrsoftware.org/isinfo.php .
Dirección del proyecto: http://nsis.sourceforge.net/Main_Page .

Ejercicios VI

Implementar un módulo con una función Fibonacci (n) que devuelva una lista de n números Tribonacci, donde n es el parámetro de la función.

Realice pruebas de la función si el módulo funciona como principal.

Implementar:

un servidor que publica un objeto distribuido, e invoca la función Fibonacci .

un cliente que usa el objeto distribuido para calcular la secuencia de Fibonacci.

Apéndices

Esta parte se centra en otras tecnologías que tienen que coexistir con Python de varias maneras. Y al final, las respuestas a los ejercicios propuestos en las partes anteriores.

Contenido:

Integración de la aplicación

Python puede utilizarse como lenguaje de scripting en muchas aplicaciones para automatizar tareas y añadir nuevas funcionalidades, o para ofrecer sus características a otro programa a través de una API o un protocolo. Muchos de estos paquetes de software son de código abierto, como BrOffice.org y Blender, por ejemplo.

Esto suele ser posible porque estos programas han adoptado una arquitectura de plugins , en la que existe una infraestructura genérica que permite vincular componentes externos a las aplicaciones.

En la mayoría de los casos, esto es posible gracias al uso de una API que proporciona el software, que Python ve como un módulo o paquete, que sólo tiene que estar en PYTHONPATH antes de poder ser utilizado. Con esto, el programa puede hacer llamadas a las rutinas de la aplicación, para usar sus recursos y comunicarse.

En otros casos, como en el de Inkscape, el programa Python actúa como un filtro, recibiendo y enviando información a la aplicación a través de la entrada estándar (stdin) y la salida (stdout).

Blender

Blender 91 es un software de modelado 3D de código abierto capaz de generar animaciones y también funciona como Game Engine (infraestructura especializada para la creación de juegos de ordenador). Además, el software cuenta con un renderizador interno y puede integrarse con renderizadores externos como los proyectos de código abierto Yafaray 92 y LuxRender 93 .

En Blender, una escena está compuesta por objetos que necesitan ser fijados en posiciones y conectados a la escena. Si el objeto no está conectado a la escena, será eliminado al final del procesamiento. Para cada sólido, se pueden configurar múltiples materiales, que pueden tener cero o más texturas.

La escena por defecto de Blender se compone de tres objetos: una cámara, un

La documentación, las fuentes y los binarios se pueden encontrar en: http://www.blender.org/.

Página del proyecto: http://www.yafaray.org/ .

Página web oficial: http://www.luxrender.net/ .

lámpara y un cubo (representado como una malla). La escala en Blender se mide en BUs (Blender Units).

Con Python es posible acceder a todas estas estructuras de Blender a través de módulos, incluyendo:

Blender : Permite abrir archivos, guardar y otras funciones relacionadas.

Objeto : operaciones con objetos 3D.

Materiales : manejo de materiales.

Texturas : manipulación de texturas.

Mundo : manipulación del entorno de la escena.

Dibujo : rutinas de la interfaz de usuario.

Nmesh : manipulación de mallas.

BGL : acceso directo a las funciones de OpenGL.

El API de Blender ofrece una variedad de texturas de procedimiento y una colección de sólidos primitivos listos para ser creados y cambiados a través del código.

Salida:

Para ejecutar el código Python en el entorno de Blender, simplemente cargue el programa en la ventana del editor de texto de Blender y utilice la

opción de ejecución en el menú.

Motor del juego

El motor de juego es un software que facilita la creación de juegos, simulando ciertos aspectos de la realidad, para permitir la interacción con

objetos en tiempo real. Para ello, debe implementar varias características que son comunes en los juegos, como las capacidades de simulación física. El objetivo principal del uso de los motores de juego es centrar el diseño del juego en el contenido, es decir, en los mapas (niveles), los personajes, los objetos, los diálogos, la banda sonora y las escenas. Es común que varios juegos utilicen el mismo motor, reduciendo así el esfuerzo de desarrollo.

Una de las características clave que ofrecen los motores de juego es la capacidad de renderizar escenas 2D o 3D en tiempo real, normalmente utilizando una biblioteca de gráficos como OpenGL, que permite realizar animaciones y efectos especiales. El componente especializado para esta función se conoce como motor de renderización .

Además, la simulación física también es esencial para que un juego represente adecuadamente los movimientos de los personajes que están siendo influenciados por la gravedad, la inercia, la fricción, la detección de colisiones y otros. El componente que realiza estos cálculos se llama Motor Físico .

Otra característica importante es la lógica, que es la forma en que el juego determina el comportamiento del entorno y de los personajes. En muchos casos, el motor del juego soporta uno o más lenguajes para describirlo.

Los motores de juego pueden incluir otros recursos importantes para ciertos tipos de juegos, como la conectividad. En el caso de los juegos masivos multijugador en línea (MMOG), que son muy complejos, la infraestructura de software es más conocida como middleware .

La popularización de los motores de juego tuvo lugar durante los años noventa, gracias a Id Software, que desarrolló juegos que definieron el género denominado FPS (First Person Shooter), juegos de acción en primera persona. Estos títulos tenían sus motores licenciados a otras compañías, que crearon otros juegos desarrollando contenidos de juegos. Paralelamente, los procesadores de vídeo fueron incorporando soporte para funciones gráficas cada vez más sofisticadas, lo que facilitó la evolución de

los motores . Id también lanzó los motores de juego de las series GPL DOOM y Quake .

Además del entretenimiento, otras áreas pueden beneficiarse de estos motores . Llamados genéricamente juegos serios , aplicaciones en las áreas de

La formación, la arquitectura, la ingeniería, la medicina y la comercialización son cada vez más populares.

Blender incluye un motor de juego genérico , que permite la creación de juegos 3D, utilizando su propia aplicación de autoría de contenidos y Python para las partes lógicas más complejas.

El motor de juego de Blender (BGE) utiliza como motor de física el proyecto, también de código abierto , llamado Bullet 94 . Con él es posible simular el comportamiento de cuerpos rígidos (como partes de máquinas), cuerpos blandos (como telas), estáticos (fijos) y cuerpos intangibles (que no se ven afectados por las colisiones).

Blender + Bullet

Picture 160 Picture 230 Picture 300

El motor hace que Blender soporte GLSL (OpenGL Shading Language), lo que le permite utilizar las características avanzadas disponibles en los últimos procesadores de video.

La lógica se define en BGE a través de Logic Bricks , que sigue un modelo basado en eventos. Los eventos se asocian a un objeto de la escena y pueden ser generados por periféricos de entrada (como el teclado y el ratón), el sistema (tiempo), el propio BGE (colisiones, por ejemplo), o los mensajes enviados por otros objetos. Cuando se detectan uno o más eventos, el BGE toma una decisión y reacciona en consecuencia.

Hay tres tipos de bricks :

Página web oficial: http://www.bulletphysics.org/ .

Sensores (sensores) que detectan los eventos.

Controladores (controladores) que correlacionan los sensores con los activadores adecuados.

Activadores (actuadores), que activan las reacciones.

En el panel Lógico , las asociaciones entre los ladrillos se pueden establecer de forma interactiva. BGE tiene varios activadores listos para realizar tareas como terminar la ejecución o cambiar la velocidad del objeto.

BGE puede evocar el código Python para responder a los eventos a través del controlador "Python". Cuando se ejecuta una función en Python, toma como argumento el controlador que hizo la llamada, por lo que es posible identificar y modificar el objeto (onwer) que posee el controlador.

Ejemplo (módulo con función de teleportación):

```
- * - coding: latin1 - * -
    Importación del módulo de interfaz del motor de juego GameLogic
    def teleport ( cont ):
    get the owner of the controller own = cont . owner
    get the scene
    scene = GameLogic . getCurrentScene ()
    # get fate
    dest = scene . getObjectList () [ 'OBr_portal' ]
    get the coordinates of destination x , y , z = dest . getPosition ()
    moves the camera to 1 BU above its own destination . setPosition ([ x , y , z + 1 ])
```

Esta función cambia la posición del objeto a un BU por encima del objeto llamado "r_portal", independientemente del lugar de la escena en que se encuentre.

GIMP 303

GIMP

GIMP 95 (GNU Image Manipulation Program) es un conocido software de código abierto que implementa varias herramientas para procesar y editar imágenes rasterizadas (con algunas características vectoriales, para el manejo de texto, por ejemplo), y algoritmos de conversión para varios formatos. Permite la manipulación de imágenes compuestas multicapa y tiene una arquitectura basada en plugins que permite implementar nuevas características.

Originalmente, los plugins se crearon en el lenguaje funcional Scheme, pero hoy en día es posible utilizar también Python, a través de una extensión llamada Gimp-Python 96 .

Un plugin hecho en Python debe seguir estos pasos:

Importar gimpfu: El módulo gimpfu define las funciones y tipos necesarios para que Python se comunique con el GIMP.

Define la función de procesamiento: La función que será usada para procesar la imagen usando el API del GIMP.

Función de registro: La función de registro () registra la función de procesamiento en la Base de Datos de Procedimientos (PDB), permitiendo al GIMP conocer la información necesaria para ejecutar el plugin .

Run main () : Rutina principal de la API.

La función de procesamiento deberá realizar algunos pasos para interactuar correctamente con el GIMP:

Recibir variables: La función toma como argumentos la imagen (imagen), la capa actual editable (dibujable), y otras que se definen en el registro de la función. El resto de parámetros se obtendrán a través de un cuadro de diálogo que se presenta al usuario antes de la ejecución.

Iniciar la transacción: Inicio de la transacción a través de la función pdb.gimp_image_undo_group_start () . La transacción puede deshacerse más tarde mediante la función "deshacer".

Procesamiento de imágenes: Cambia la imagen o la capa a través de las funciones definidas en la API.

Ventana de opciones:

Ejemplo de pasos para la generación de imágenes:

El GIMP también permite que los plugins se ejecuten desde la línea de comandos usando los parámetros "—no-interface—batch".

El script debe estar en una carpeta donde el GIMP pueda encontrarlo. Para el GIMP 2.6, la carpeta de plugins de usuario está en .gimp-2.6 / plugins debajo del directorio principal del usuario . Además, la extensión requiere que PyGTK y sus dependencias estén instalados.

Inkscape 307

Inkscape

El editor de imágenes vectoriales Inkscape 97 permite el uso de Python como lenguaje de scripts para crear extensiones. La aplicación utiliza SVG como su formato nativo e implementa varias características proporcionadas en el estándar.

Las extensiones para Inkscape son principalmente para implementar filtros y efectos. Mientras que otras aplicaciones (como Blender) presentan una API en forma de módulos para el intérprete, Inkscape pasa argumentos a través de la línea de comandos y transfiere información a través de la entrada y salida del sistema operativo estándar, de forma similar a las utilidades de procesamiento de textos que se encuentran en los sistemas UNIX. Con ello, la extensión sólo tiene acceso a los elementos que forman parte del documento, no a la interfaz gráfica de la aplicación. Cualquier interacción con el usuario durante la ejecución depende de la extensión.

La creación y manipulación de las estructuras de datos se realiza mediante el uso de XML, tal y como se especifica en el formato SVG, permitiendo así el uso de módulos como ElementTree .

Para simplificar el proceso, Inkscape proporciona un módulo llamado inkex , que define las estructuras básicas para las extensiones. Con este módulo, se pueden crear nuevas extensiones por herencia de una clase llamada Effect .

Ventana con los parámetros de la extensión:

Ejemplo de salida (cantidad 100, anchura 600 y altura 200):

Inkscape 311

Tanto el programa como el archivo de configuración deben estar en la carpeta de extensiones (comparta las extensiones \, dentro de la carpeta de instalación, para la versión de Windows) para ser encontrados por la aplicación, y el nombre del archivo de configuración debe tener una extensión de ".inx". .

LibreOffice.org

BrOffice.org 98 es un conocido paquete de automatización de oficina de código abierto que incluye un editor de texto, una hoja de cálculo y otras aplicaciones. Además, BrOffice.org también soporta Python (entre otros lenguajes):

Como un lenguaje de macros, permitiendo la automatización de tareas.

Para la construcción de extensiones (add ons).

En un servicio de servicio de conexiones, a través de una API llamada Universal Network Objects (UNO).

Ejemplo de macro:

```
- * - coding: latin1 - * -
El macro debe ser ejecutado desde
LibreOffice.org Calc
def plan ():
"" "
Llenar la hoja de calculo
"" "
Get the document for the doc = XSCRIPTCONTEXT script context .
getDocument ()
Integración con otros idiomas
```

También puedes integrar Python con Fortran usando la utilidad f2py, que es parte del proyecto NumPy.

Bibliotecas compartidas

A partir de la versión 2.5, Python incorporó el módulo ctypes , que implementa tipos compatibles con los tipos utilizados por el lenguaje C y permite evocar funciones de biblioteca compartidas.

El módulo proporciona varias formas de evocar funciones. Las funciones que siguen la convención de llamadas stdcall , como la API de Windows, pueden ser accedidas a través de la clase windll . Dynamic-link library (DLL) es la implementación de las bibliotecas compartidas que se utilizan en Windows.

Ejemplo con windll :

# - * - coding: latin1 - * -
import ctypes
Llamando al buzón de mensajes de Windows
Los argumentos son: ventana padre, mensaje,
título de la ventana y tipo de ventana.
La función devuelve un número entero, que

corresponde a qué botón fue presionado
= ctypes . windll . user32 . MessageBoxA (None , 'DLL Test!' , 'Message' , 0)
El resultado indica qué botón se pulsó print i

Para las funciones que siguen la convención de llamada cdecl utilizada por la mayoría de los compiladores de C, existe la clase cdll . Para los pasajes de argumentos por referencia se debe crear una variable que actúe como un buffer para recibir los resultados. Esto es necesario para recibir cadenas de texto, por ejemplo.

Ejemplo con cdll y buffer:

```
# - * - coding: latin1 - * -
import ctypes
msvcrt es la biblioteca con más funciones
Patrones de lenguaje C en Windows
Windows pone automáticamente
la extensión del archivo
clib = ctypes . cdll . msvcrt
Crea un búfer para recibir el resultado.
La referencia del buffer se pasará a
la función, que llena el buffer con el resultado s = ctypes . c_buffer ( '\
000' , 40 )
sscanf () es una función que extrae valores
de una cuerda como una máscara
clib . sscanf ('Probando sscanf! \ n' ,
¡Probando! \ N' , s )
Muestra el resultado de la impresión.
```

También puede llamar funciones de biblioteca compartida en Linux:

```
# - * - codificación: latín1 - * -
```



```
import ctypes
Cargar la biblioteca C por defecto en Linux
La extensión del archivo necesita ser aprobada
para la función LoadLibrary ()
clib = tipos . cdll . LoadLibrary ( "libc.so.6" )
Crea un buffer para recibir el resultado s = ctypes . c_buffer ( '\ 000' , 40
)
Llamar la función de Sprintf
clib . sprintf ( s , 'Testing% s \ n' , 'sprintf!' )
Muestra el resultado de la impresión.
```

A través de las bibliotecas compartidas, es posible utilizar el código desarrollado en otros idiomas de manera sencilla.

Escritura Dinámica

La escritura en Python es fuerte, lo que significa que el intérprete comprueba si las operaciones son válidas y no restringe automáticamente los tipos incompatibles. Para realizar la operación entre tipos incompatibles, debe convertir explícitamente el tipo de la variable o variables antes de la operación.

Compilación e Interpretación

Por defecto, el intérprete compila el código y almacena el código de bytes en el disco para que la próxima vez que lo ejecute no tenga que volver a compilar el programa, reduciendo el tiempo de carga en la ejecución. Si se cambian los archivos fuente, el intérprete se encargará de regenerar el código de bytes automáticamente, incluso usando el Shell interactivo. Cuando se invoca un programa o módulo, el intérprete realiza el análisis del código, convierte a símbolos, compila (si no hay código de bytes actualizado en el disco), y se ejecuta en la máquina virtual Python.

El código de bytes se almacena en archivos con la extensión ".pyc" (código de bytes normal) o ".pyo" (código de bytes optimizado). El bytecode también puede ser empaquetado junto con el intérprete en un ejecutable para un fácil despliegue de la aplicación, eliminando la necesidad de instalar Python en cada computadora.

Modo Interactivo

El intérprete de Python puede utilizarse de forma interactiva, en el que se escriben líneas de código en un prompt similar al Shell del sistema operativo.

Para evocar el modo interactivo sólo hay que ejecutar el intérprete (si está en camino):

Cultura

La comunidad de usuarios de Python ha creado algunas expresiones para referirse a temas relacionados con el lenguaje. En esta jerga, el término Pythonic se utiliza para indicar que algo es compatible con los supuestos de diseño de Python, y Unpythonic significa lo contrario. El usuario del lenguaje es llamado desde el pitonista.

Los objetivos del proyecto fueron resumidos por Tim Peters en un texto llamado Zen de Python, que está disponible en el propio Python a través del comando:

importar esto

El texto hace hincapié en la postura pragmática del Dictador Benévolo por la Vida (BDFL), como se conoce a Guido en la comunidad pitoniana.

Las propuestas de mejora del lenguaje se denominan Propuestas de Mejora de la Pitón (PEP), que también sirven de referencia para las nuevas características que se implementen en el lenguaje.

Además del sitio oficial, otras buenas fuentes de información sobre el idioma son PythonBrasil, el sitio de la comunidad python en el Brasil, con mucha información en portugués, y Python Cookbook, un sitio que almacena "recetas": pequeñas porciones de código para realizar tareas específicas.

Sintaxis

Un programa Python se compone de líneas, que pueden continuar en las siguientes líneas utilizando el carácter de barra invertida (\) al final de la línea o paréntesis, paréntesis o corchetes en las expresiones que utilizan tales caracteres.

El carácter # marca el comienzo del comentario. Cualquier texto después de # será ignorado hasta el final de la línea, excepto los comentarios funcionales.

Los comentarios funcionales se utilizan para:

Cambiar la codificación del archivo fuente del programa añadiendo un comentario con el texto "# - * - codificación": <codificación> - * # -" al principio del archivo, donde <codificación> es la codificación del archivo (normalmente latin1 o utf -8). Se requiere cambiar la codificación para soportar caracteres no ingleses en el código fuente del programa).

Ejemplo de comentarios funcionales:

#! / usr / bin / env python

Una línea de código que muestra el resultado 7 veces 3 imprime 7 * 3

Salida:

21

Ejemplos de líneas discontinuas:

Una línea cortada por una barra invertida = 7 * 3 + \

5 / 2

Una lista (rota por una coma)= ['a', 'b', 'c', 'd', 'e']

Una función rota por comas llamada c = rango (1,11)

imprimir todo en la pantalla

imprimir a, b, c

Salida:

23 ['a', 'b', 'c', 'd', 'e'] [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

El comando de impresión inserta espacios entre las expresiones que se reciben como parámetro y un carácter de nueva línea al final, a menos que reciba una coma al final de la lista de parámetros.

Bloques

En Python, los bloques de código están delimitados por el uso de indentación, que debe ser constante en el bloque de código, pero se considera una buena práctica mantener la coherencia en todo el proyecto y evitar la mezcla de tabulaciones y espacios.

La línea que precede al bloque siempre termina con dos puntos (:) y representa una estructura de control del lenguaje o una declaración de una nueva estructura (una función, por ejemplo).

Ejemplo:

Inicio del Programa

Dos Puntos requeridos

Inicio de un bloque

Otro Bloque

Fin e los dos Bloques

Fin del Programa

For i in list 234, 654, 378, 798: for i in [234, 654, 378, 798]:

Si el resto dividido entre 3 es igual a cero: if i % 3 == 0 :

Print...

print i, '/ 3 =', i / 3

Salida:

234/3 = 78

654/3 = 218

378/3 = 126

798/3 = 266

El operador “%” calcula el módulo (residuo de la división)

Objetos

Python es un lenguaje orientado a los objetos, por lo que las estructuras de datos tienen atributos (los datos en sí) y métodos (rutinas asociadas a los datos). Se accede tanto a los atributos como a los métodos mediante el uso de dot (.).

Para mostrar un atributo:

print object. attribute

Para ejecutar un método:

object. method (argumentos)

Incluso un método sin argumentos necesita paréntesis:

object. method ()

El punto también es utilizado para acceder a las estructuras del modulo que fueron importadas por el programa.

Control de Flujo

Introduzca la temperatura: 23

Normal

Donde: "Introducir temperatura:" es el mensaje que indica que el programa espera la entrada, "23" es la entrada introducida, y "Normal" es la respuesta del programa.

Si el bloque de código consiste en una sola línea, se puede escribir después de los dos puntos:

if temp < 0 : print 'Freezing...'

A partir de la versión 2.5, Python soporta la expresión:

<variable> = <value 1> if <condition> else <value 2>

Where <variable> recibira <value 1> if <condition> es verdadero y <value 2> el caso contrario.

Lazos

Los ciclos (loops) son estructuras repetidas, utilizadas comúnmente para procesar colecciones de datos, como líneas de un archivo o registros de una base de datos, que deben ser procesados por el mismo bloque de código.

For

Es la estructura repetitiva más comúnmente usada en Python. La declaración acepta no sólo secuencias estáticas sino también secuencias generadas por iteradores. Los iteradores son estructuras que permiten iteraciones, es decir, el acceso a elementos de una colección de elementos de forma secuencial.

Durante la ejecución de un bucle, la referencia apunta a un elemento de la secuencia. En cada iteración, la referencia se actualiza, de modo que el bloque de código debe procesar el elemento correspondiente.

Tipos

Hay varios tipos de datos simples predefinidos en Python, como:

Números (enteros, reales, complejos...), Texto.

También hay tipos que funcionan como colecciones. Los principales son:

Lista,

Tupla,

Diccionario.

Los tipos de Python pueden ser:

Cambiables: permiten cambiar el contenido de las variables.

Inmutable: no permiten que se cambie el contenido de las variables.

En Python, los nombres de las variables son referencias, que pueden ser cambiadas en tiempo de ejecución.

Los tipos y las rutinas más comunes se implementan en forma de builtins, lo que significa que siempre están disponibles en tiempo de ejecución sin necesidad de importar ninguna biblioteca.

Números

Python tiene una serie de operadores definidos para manipular los números, a través de cálculos aritméticos, operaciones lógicas (que comprueban si una condición es verdadera o falsa), o procesamiento de bits (donde los números se tratan en forma binaria).

Operaciones aritméticas:

Suma (+).

Diferencia (-).

Multiplicación (*).

División (/): Entre dos números enteros funciona igual que la división de números enteros. En otros casos, el resultado es real.

División entera (//): El resultado se trunca al siguiente número entero inferior, incluso cuando se aplica a números reales, pero en este caso, el resultado también será real.

Módulo (%): Devuelve el resto de la división.

Potencia (**): puede utilizarse para calcular la raíz a través de los exponentes fraccionarios (ejemplo: 100 ** 0,5).

Positivo (+).

Negativo (-).

Los símbolos pueden ser usados para mostrar los números en varios formatos.

Ejemplos:

Ceros al principio

print 'Now it's% 02d:% 02d.' % (16, 30)

Real (el número después del punto controla los decimales)

print "Percentage:%.1f %% Exponential:.% 2e ' % (5.333, 0.00314)

Octal and hexadecimal

print 'Decimal:% d, Octal:% o, Hexadecimal:% x' % (10, 10, 10)

Salida:

Ahora son las 4:30 pm.

Porcentaje: 5.3%, Exponencial: 3.14e-03

Decimal: 10, Octal: 12, Hexadecimal: a

A partir de la versión 2.6, se dispone de otra forma de interpolación además del operador "%", el método de la cadena y la función llamada formato ().

Ejemplos:

musicians = [('Page', 'guitarist', 'Led Zeppelin'), ('Fripp', 'guitarist', 'King Crimson')]

Parámetros identificados en orden msg = '{0} is {1} from {2}'

for name, function, band in musicians :

print (msg. format (name, function, band))

Parámetros Identified by Name

msg = '{greeting}, are {time: 02d}: {minute: 02d}'

print msg. format (greeting = 'Good morning', hour = 7, minute = 30)

Builtin format () function

print 'Pi =', format (3.14159, '.3e')

Salida:

Page is a guitarist for Led Zeppelin

Fripp is a guitarist for King Crimson

Good morning, it's 07:30

Pi = 3.142e + 00

La función de formato () puede utilizarse para dar formato a un solo dato a la vez.

Se pueden obtener rebanadas (trozos) de cadenas colocando paréntesis después de la cadena de índice.

Los índices en Python:

Comienzan en cero.

Cuentan desde el final si son negativos.

Se pueden definir como estiramientos, en la forma [inicio: fin + 1: intervalo]. Si no se fija el inicio, se considerará cero. Si no se establece el final + 1, se considera el tamaño del objeto. El intervalo (entre caracteres), si no se establece, será 1.

Puede invertir las cadenas utilizando un intervalo negativo:

print 'Python' [:: - 1]

Shows: nohtyP

En el módulo de cadenas se han implementado varias funciones para el manejo de texto.

importación del módulo de cadenas de texto importación de cadenas de texto

El alfabeto

a = string. ascii_letters

Rotando el alfabeto un carácter a la izquierda b = a [1 :] + a [0]

La función maketrans () crea una tabla de traducción entre los caracteres de las dos cadenas que recibió como parámetro.

Los caracteres que faltan en las tablas se copiarán en la salida.

tab = string. maketrans (a, b)

El mensaje...

msg = " 'This text will be translated..

sera muy extrano.

" '

La función translate () utiliza la tabla de traducción creada por maketrans () para traducir una cadena de impresión. translate (msg, tab)

Salida:

Fttf ufyup tfsá usbevAjep..

Wbj gjdbs cfn ftusboip.

El módulo también implementa un tipo llamado Template, que es una plantilla de cadena que puede ser rellenada a través de un diccionario. Los identificadores están iniciados con el signo de dólar ($) y pueden estar rodeados de claves para evitar confusiones.

Ejemplo:

importar el módulo de la cadena importar la cadena

Crear una plantilla de cadena

st = string. Template ('$ warning happened on $ when')

Llene el modelo con un diccionario

= st. substitute ({ 'warning' : 'Power outage', 'when' : ' April 3, 2002' })

Muestra:

Lack of electricity happened on April 3, 2002 print s

Puedes usar cadenas mutables en Python a través del módulo UserString, que define el tipo de MutableString:

Importar el modulo ing UserString

import UserString = UserString. MutableString ('Python') s [0] = 'p'

print s # show "python"

Las cadenas cambiantes son menos eficientes que las inmutables porque son más complejas (en términos de estructura), lo que resulta en un mayor consumo de recursos (CPU y memoria).

Para utilizar ambos métodos, es necesario pasar por una codificación compatible. Los más utilizados con el idioma portugués son “latin1” and “utf8”.

Listas

Las listas son colecciones heterogéneas de objetos, que pueden ser de cualquier tipo, incluyendo otras listas.

Las listas en Python son cambiantes y pueden ser modificadas en cualquier momento. Las listas pueden ser cortadas como cadenas, pero como las listas son cambiantes, se pueden hacer asignaciones a los elementos de la lista.

Sintaxis:

list = [a, b,..., z]

Lista Comun de Operaciones:

Una nueva lista: Brit Progs from the 70s progs = ['Yes', 'Genesis', 'Pink Floyd', 'ELP']

La función enumerar () devuelve una tupla de dos elementos para cada iteración: un número de secuencia y un elemento de la secuencia correspondiente.

La lista tiene el método pop () que facilita la implementación de filas y columnas:

list = ['A', 'B', 'C']

print 'list:', list

La lista vacía evalúa a la lista de falsos mientras que la lista:

En las filas, el primer ítem es el primero en salir pop (0) elimina y devuelve el primer ítem de impresión 'Exit', list. pop (0), 'missing', len (list)

Mas listas de Ítems

list + = ['D', 'E', 'F']

print 'list:', list

while list:

En las columnas, el primer artículo es el último que sale pop () elimina y devuelve el último artículo impreso 'Out', list. pop (), 'missing', len (list)

Salida:

list: ['A', 'B', 'C']

Left A, 2 left

Left B, 1 left

Left C, 0 left

list: ['D', 'E', 'F']

Left F, 2 left

Left E, 1 left

Left D, 0 left

Las operaciones de clasificación (ordenar) y anulación (revertir) se realizan en la lista; por lo tanto, no se generan nuevas listas.

Uplas

Similar a las listas, pero sin cambios: No se puede añadir, borrar o asignar elementos.

Sintaxis:

tuple = (a, b,..., z)

Los paréntesis son opcionales.

Particularidad: la tupla con un solo elemento, se representa como:

t1 = (1,)

Los elementos de una tupla pueden ser referenciados de la misma manera que los elementos de una lista:

first_element = tuple [0]

Las listas pueden convertirse en tuplas:

tuple = tuple (list)

Y las tuplas pueden ser convertidas en listas:

list = list (tuple)

Aunque la tupla puede contener elementos mutables, estos elementos no pueden asignarse, ya que esto modificaría la referencia al objeto.

Ejemplo (utilizando el modo interactivo):

t = ([1, 2], 4)

t [0]. append (3)

t

([1, 2, 3], 4)

>>> t [0] = [1, 2, 3]

Rastreo (última llamada más reciente):

File "<input>", line 1, in ?

TypeError: El objeto no soporta la asignación de posiciones

>>>

Las tuplas son más eficientes que las listas convencionales porque consumen menos recursos computacionales (memoria), ya que son estructuras más simples, como cuerdas inmutables sobre cuerdas mutables.

Otros tipos de secuencias

Python también proporciona entre los constructores:

conjunto: secuencia mutable unívoca (sin repeticiones) no ordenada.

frozenset: secuencia inalterable no ordenada y sin cambios.

Cuando una lista se convierte en conjunto, las repeticiones se descartan.

Diccionarios

Un diccionario es una lista de asociaciones compuesta por una sola clave y las estructuras correspondientes. Los diccionarios son modificables, como las listas.

La clave debe ser de tipo inmutable, se suelen utilizar cadenas, pero también pueden ser tuplas o tipos numéricos. Los elementos del diccionario

pueden ser tanto cambiantes como no cambiantes. El diccionario Python no garantiza la clasificación de las claves.

Sintaxis:

dictionary = { 'a' : a, 'b' : b,..., 'z' : z }

Estructura:

{ 'Fractal' : 'IFS', 'Reed' : 'Green', 'Limits' : (640, 480), (0, 0): (42, 35)}

Ejemplo de Diccionario:

dic = { 'name' : 'Shirley Manson', 'band' : 'Garbage' }

Elementos de acceso:

print dic ['name']

Elementos de Adicion:

dic ['album'] = 'Version 2.0'

Borrar un elemento de un diccionario:

del dic ['album']

Obtener los Ítems, claves y valores:

items = dic. items ()

Claves = DIC. keys ()

Valores = DIC. values ()

Salida:

Yes => ['Close To The Edge', 'Fragile']

ELP => ['Brain Salad Surgery']

Genesis => ['Foxtrot', 'The Nursery Crime']

King Crimson => ['Red', 'Discipline']

Ejemplo de matriz dispersa:

Matriz dispersa implementada con el diccionario

La matriz dispersa es una estructura que sólo almacena los valores que existen en la matriz

dim = 6, 12

mat = {}

Las tuplas no cambian

Cada tupla representa una posición en la matriz [3, 7] = 3

mat [4, 6] = 5 mat [6, 3] = 7 mat [5, 4] = 6 mat [2, 9] = 4 mat [1, 0] = 9

for lin in range (dim [0]):

for col in range (dim [1]):

Método de obtención (clave, valor)

devuelve el valor de la clave en el diccionario o si la clave no existe,
devuelve el segundo argumento
print mat. get ((lin, col), 0),
print
Salida:
000000000000
900000000000
000000000400
000000030000
000000500000
000060000000

Obtener la matriz dispersa:

Array in the form of string array = " '0 0 0 0 0 0 0 0 0 0 0 0 0
900000000000 000000000400 000000030000 000000500000
000060000000 " '
mat = {}
# Romper la Matriz en Líneas:
for lin, line in enumerate (array. splitlines ()):
# Envolver las líneas en columnas
for col, column in enumerate (row. split ()):
column = int (column)
Colocar la columna en el resultado si no es cero
if column :
mat [lin, col] = column
print mat
Sume uno en las dimensiones porque el conteo comienza en cero.
"Tamaño de la matriz completa":', (lin + 1) * (col + 1) print 'Sparse array size:', len (mat)
Salida:
{(5,4): 6, (3,7): 3, (1,0): 9, (4,6): 5, (2,9): 4}
Tamaño del Arreglo Completo: 72
Tamaño de la Matriz Dispersa: 5

La matriz dispersa es una buena solución de procesamiento para las estructuras en las que la mayoría de los elementos permanecen vacíos, como las hojas de cálculo.

Verdadero, falso y nulo

En Python, el tipo booleano (bool) es una especialización entera (int). True se llama True y es igual a 1, mientras que False se llama False y es igual a cero.

Los siguientes valores se consideran falsos:

False.

Ninguno (null).

0 (cero).

" (cadena vacía)

[] (lista vacía).

() (tupla vacía).

{} (diccionario vacío).

Otras estructuras con tamaño igual a cero.

Todos los demás objetos fuera de esta lista se consideran verdaderos.

El objeto None, que es del tipo NoneType, Python es nulo y es calificado como falso por el intérprete.

Los operadores booleanos

Con operadores lógicos, es posible construir condiciones más complejas para controlar las desviaciones condicionales y los bucles.

Los operadores booleanos en Python son: y, o, no, es y en.

y : Devuelve un valor verdadero si y sólo si recibe dos expresiones que sean verdaderas.

o : Devuelve un valor falso si y sólo si recibe dos expresiones que son falsas.

not : devuelve falso si recibe una expresión verdadera y viceversa.

is : devuelve verdadero si se le dan dos referencias al mismo objeto y falso en caso contrario.

in : devuelve verdadero si recibe un objeto y una lista, y el objeto aparece una o más veces en la lista y falso en caso contrario.

El valor resultante se calcula en la operación y de la siguiente manera: si la primera expresión es verdadera, el resultado es la segunda expresión, si no la primera. Para el operador o , si la primera expresión es falsa, el resultado será la segunda expresión. En caso contrario, será la primera. Para los otros operadores, el retorno será bool (Verdadero o Falso).

Ejemplos:

print 0 and 3 # Muestra 0

print 2 and 3 # Muestra 3

print 0 or 3 # Muestra 3

print 2 or 3 # Muestra 2

print not 0 # Muestra True

print not 2 # Muestra False

print 2 in (2, 3) # Muestra True

print 2 is 3 # Muestra False

Además de los operadores booleanos, están las funciones all (), que devuelven verdadero cuando todos los elementos son verdaderos en la secuencia utilizada como parámetro, y any (), que devuelve verdadero si cualquier elemento es verdadero.

Módulos

Para Python, los módulos son archivos fuente que pueden ser importados a un programa. Pueden contener cualquier estructura Python y se ejecutan al ser importados. Se compilan cuando se importan por primera vez y se almacenan en un archivo (con extensión ".pyc" o ".pyo"), tienen su propio espacio de nombres y soportan Doc Strings. Son objetos Singleton (sólo se carga una instancia en la memoria, que está disponible globalmente para el programa).

Los módulos son localizados por el intérprete a través de la lista de carpetas de PYTHONPATH (sys.path), que normalmente incluye primero la carpeta actual. Los módulos se cargan a través de la declaración de importación. Por lo tanto, cuando se utiliza alguna estructura de módulo, es necesario identificar el módulo. Esto se denomina importación absoluta.

importarlos

print os. name

También se pueden importar módulos relativamente:

Desde el os import name

Imprimir nombre

El carácter "*" puede ser usado para importar todo lo que se define en el módulo:

Desde el os import *

Imprimir nombre

Al evitar problemas como la ofuscación de las variables, la importación absoluta se considera una mejor práctica de programación que la

importación relativa.

Ejemplo de módulo:

Calc.py file

Funcion definida en def media module (list):

return float (sum (list)) / len (list)

Ejemplo del uso del modulo:

Importar el calculo import calc module

= [23, 54, 31, 77, 12, 34

Llama la function definida en calc print calc. media (l)

Salida:

38.5

El módulo principal de un programa tiene la variable __nombre__ igual a "__main__", así que puedes probar si el módulo es el principal usando:

if __name__ == "__main__" :

Aquí el código sólo será ejecutado

si este es el módulo principal

no cuando es importado por otro programa

Esto hace que sea fácil convertir un programa en un módulo.

Dividir los programas en módulos facilita la reutilización y la búsqueda de fallos en el código.

Name Scope

El alcance de los nombres en Python se mantiene a través de los Namespaces, que son diccionarios que relacionan los nombres de los objetos (referencias) y los objetos mismos.

Normalmente, los nombres se definen en dos diccionarios, que pueden consultarse a través de las funciones locales () y globales (). Estos diccionarios se actualizan dinámicamente en tiempo de ejecución.

Un espacio de nombres es un ámbito de definición de estructuras.

Se trata de variables globales.

Var_3 fue eclipsada como lo fue

(re) definido en el ámbito local.

Estas son variables locales.

Las variables globales pueden quedar eclipsadas por las locales (ya que el ámbito local se consulta antes que el ámbito global). Para evitar esto, debe declarar la variable como global en el ámbito local.

Ejemplo:

Aunque los diccionarios devueltos por locales () y globales () pueden cambiarse directamente, esto debe evitarse ya que puede tener efectos indeseables.

```
Somali def ( list ):
"" "
La suma de las listas es recursiva
Ponga el resultado como global
"" "
Suma global
for item in list :
if type ( item ) is list : # If item type is Somalist list ( item )
else :
sum + = item
sum = 0
Somali ([[ 1, 2 ], [ 3, 4, 5 ], 6 ])
print sum # 21
```

El uso de variables globales no se considera una buena práctica de desarrollo, ya que dificulta la comprensión del sistema, por lo que es mejor evitar su uso. Y ofuscar las variables también.

Paquetes

Los paquetes son carpetas que el intérprete identifica por la presencia de un archivo llamado "__init__.py". Los paquetes actúan como colecciones para organizar los módulos de forma jerárquica.

Se pueden importar todos los módulos del paquete utilizando la declaración de importación de nombre_de_paquete *.

El archivo "__init__.py" puede estar vacío o contener el código de inicialización del paquete o establecer una variable llamada __all__, se importará una lista de módulos del paquete cuando se utilice "*". Sin el archivo, Python no identifica la carpeta como un paquete válido.

Biblioteca Estándar

A menudo se dice que Python viene con "baterías incluidas", en referencia a la vasta biblioteca de módulos y paquetes que se distribuyen con el intérprete. Algunos módulos importantes de la biblioteca por defecto:

Matemáticas: matemáticas, cmath, decimal y aleatorio.

Sistema: os, glob, shutils y subproceso.

Hilos: threding.

Persistencia: pickle y cPickle.
XML: xml.dom, xml.sax y elementTree (a partir de la versión 2.5).
Configuración: ConfigParser y optparse.
Hora: hora y fecha.
Otros: sys, logging, traceback, types y timeit.

Matemáticas

Además de los tipos numéricos incorporados del intérprete, en la biblioteca estándar de Python hay varios módulos dedicados a la realización de otros tipos y operaciones matemáticas.

El módulo matemático define funciones logarítmicas, de exponenciación, trigonométricas, hiperbólicas y de conversión angular, entre otras. El módulo de cmath, por otro lado, implementa funciones similares pero hechas para procesar números complejos.

Las fracciones pueden ser inicializadas de varias maneras: como una cadena, como un par entero o como un número real. El módulo también tiene una función llamada gcd (), que calcula el mayor divisor común (MDC) entre dos enteros.

Archivos y I / O

Los archivos en Python están representados por objetos de tipo de archivo 18, que proporcionan métodos para diversas operaciones de archivo. Los archivos pueden abrirse para leer ('r', que es el valor predeterminado), escribir ('w') o añadir ('a'), en modo texto o binario ('b').

En Python:
sys.stdin representa la entrada estándar.
sys.stdout representa la salida estándar.
sys.stderr representa la salida estándar de errores.

La entrada, la salida y el error estándar son tratados por Python como archivos.

La referencia abierta apunta al archivo.

Abre la entrada en modo de lectura y los otros en modo de escritura.

Los objetos de archivo también tienen un método de búsqueda (), que permite ir a cualquier posición del archivo.

En la versión 2.6, está disponible el módulo io, que implementa por separado las operaciones de archivo y las rutinas de manipulación de texto.

Expresiones Regulares

Una expresión regular es una forma de identificar patrones en las cuerdas. En Python, el módulo re proporciona un analizador que permite el uso de tales expresiones. Los patrones definidos por caracteres que tienen un significado especial para el analizador.

Bibliotecas de Terceros

Hay muchas bibliotecas escritas de terceros disponibles para Python, consistentes en paquetes o módulos, que implementan muchas características más allá de la biblioteca estándar.

En general, las bibliotecas se distribuyen de las siguientes maneras:

Paquetes de Distutils.

Paquetes para administradores de paquetes de sistemas operativos.

Instaladores

Huevos de Python.

Los paquetes que usan el módulo de distutils, que se distribuye con Python, son muy populares. Los paquetes se distribuyen en archivos comprimidos (normalmente ".tar.gz", ".tar.bz2" o ".zip"). Para instalarlos, hay que descomprimir el archivo, entrar en la carpeta descomprimida y finalmente ejecutar el comando:

python setup.py install

Que el paquete se instalará en la carpeta "site-packages" en Python.

Los administradores de paquetes del sistema operativo a menudo trabajan con sus propios formatos de paquetes, como ".deb" (Debian Linux) o ".rpm" (RedHat Linux). La forma de instalar los paquetes depende del gestor utilizado. La gran ventaja es que el gestor de paquetes se encarga de las dependencias y las actualizaciones.

Los programas instaladores no son más que ejecutables que instalan la biblioteca. Normalmente se utilizan en el entorno de Windows y pueden ser desinstalados por el Panel de Control.

Python Egg es un formato de paquete (con la extensión ".egg") que es administrado por easy_install, una utilidad que forma parte del proyecto setuptools 19.

Ruby Gems se está convirtiendo poco a poco en el estándar de facto para la distribución de librerías Python.

El programa busca la última versión del paquete en el Índice de paquetes Python (PYPI 20), el repositorio de paquetes Python, y también intenta instalar las dependencias que necesite. Los paquetes Python Eggs se pueden instalar mediante el comando:

easy_install package_name

El script easy_install está instalado en la carpeta de scripts de Python.

Dirección: http://pypi.python.org/pypi.

El manejo de excepciones puede tener un bloque más, que se ejecutará cuando no haya ninguna excepción y un bloque finalmente, se ejecutará de todos modos, habiendo sido una excepción. Se pueden definir nuevos tipos de excepción a través de la herencia de la clase de excepción.

A partir de la versión 2.6, está disponible la sentencia with, que puede anular la combinación try / finally en muchas situaciones. Con 'with', podemos definir un objeto que se utilizará durante la ejecución del bloque. El objeto debe soportar el protocolo de gestión de contexto, lo que significa que debe tener un método __entrar __ (), que se ejecuta al principio del bloque, y otro llamado __salir __ (), que se evoca al final del bloque.

Clases

En Python, hay dos tipos de clases, llamadas estilo antiguo y estilo nuevo. Las clases de estilo nuevo se derivan de la clase de objeto y pueden utilizar nuevas características de las clases de Python, como propiedades y metaclases. Las propiedades son atributos en tiempo de ejecución calculados mediante métodos, mientras que las metaclases son clases que generan clases con las que se puede personalizar el comportamiento de las clases. Las clases de estilo antiguo son un legado de versiones anteriores de Python, mantenidas para garantizar la compatibilidad con el código heredado.

Sintaxis:

```
class Class ( supcl1, supcl2 ):
"" "
Esto es una clase
"" "
clsvar = []
def __init__ ( self, args ):
"" "
Inicializador de Clases
"" "
< code block >
def __done__ ( self ):
"" "
Destructor de clase
```

```
""" "
< code block >
def method ( self, params ):
""" "
Método de Objeto
""" "
< code block >
@classmethod
def cls_method ( cls, params ):
""" "
Metodo de Clase
""" "
< code block >
@staticmethod
def this_method ( params ):
""" "
Metodo Estatico
""" "
< code block >
obj = Class ()
obj. method ()
Class. cls_method ()
Class. this_method ()
```

Los métodos de objetos pueden utilizar atributos y otros métodos de objetos. La auto variable, que representa el objeto y también necesita ser pasada explícitamente. El nombre self es una convención, como cls, y puede ser cambiado por cualquier otro nombre, pero se considera una buena práctica mantener el nombre.

Los métodos de clase sólo pueden utilizar atributos y otros métodos de clase. El argumento cls representa la clase en sí misma, debe ser pasado explícitamente como el primer parámetro del método.

Los métodos estáticos son aquellos que no tienen conexión con los atributos del objeto o de la clase. Funcionan como funciones comunes.

En Python no existen variables y métodos privados (a los que sólo se puede acceder desde el propio objeto). En su lugar se utiliza una convención, el uso de un nombre que comienza con un guión bajo (_) debe

considerarse parte de la implementación interna del objeto y está sujeto a cambios sin previo aviso.

Herencia Simple

La herencia es un mecanismo que proporciona la orientación a los objetos, con el fin de facilitar la reutilización del código. La idea es que las clases se construyen formando una jerarquía.

La nueva clase puede implementar nuevos métodos y atributos y heredar los métodos y atributos de la clase antigua (que también puede haber heredado de clases anteriores), pero estos métodos y atributos pueden ser anulados en la nueva clase.

La forma común de herencia se denomina herencia simple, en la que la nueva clase se deriva de una sola clase existente, pero se pueden crear múltiples clases derivadas mediante la creación de una jerarquía de clases.

Para encontrar métodos y atributos, la jerarquía se sigue de abajo hacia arriba, de forma similar a la búsqueda de los espacios de nombres locales y globales.

Reiniciando y/o creando métodos y/o atributos.

En las clases de estilo antiguo, la resolución comienza con la clase más a la izquierda y baja hasta el final de la jerarquía y luego pasa a la rama derecha.

Ya en las nuevas clases de estilo, la resolución se hace desde la izquierda, descendiendo hasta encontrar la clase común entre los caminos dentro de la jerarquía. Cuando se encuentra una clase común, la búsqueda va por el camino correcto. Cuando los caminos se agotan, el algoritmo procede a la clase común y repite el proceso.

En la jerarquía de clases del ejemplo, el MRO para la clase anfibia será:

[<class '__main __. Amphibious'>,
<class '__main __. Car'>,
<class '__main __. Earth'>,
<class '__main __. Boat'>,
<class '__main __. Aquatic'>,
<type 'object'>]

La herencia múltiple es una característica controvertida porque su uso puede hacer que el proyecto sea confuso y oscuro.

Serialización

La forma más simple y directa de persistencia se llama serialización 25 y consiste en escribir en el disco un volcado del objeto, que puede ser cargado posteriormente. En Python, la serialización se implementa de muchas maneras, siendo la más común a través del módulo llamado pickle.

Ejemplo de serialización:

El programa intenta recuperar el diccionario de configuración usando el objeto de archivo “setup.pkl”.

Si tiene éxito, imprime el diccionario.

Si falla, crea una configuración predeterminada y la guarda en “setup.pkl”.

Los módulos estándar de la biblioteca incluyen otros módulos de persistencia, como:

cPickle: versión más eficiente de pickle, pero no puede ser subclasificado.

shelve: proporciona una clase de objetos persistentes similares al diccionario.

Hay marcos Python de terceros que ofrecen formas más persistentes de persistencia, como ZODB.

Todas estas formas de persistencia almacenan datos en formas binarias, que no son directamente legibles por los humanos.

Para almacenar datos en forma de texto, hay módulos Python para leer y escribir estructuras de datos en formatos:

JSON 26 (JavaScript Object Notation).

YAML 27 (YAML Ain't a Markup Language).

XML 28 (Extensible Markup Language).

ZODB

La Base de Datos de Objetos Zope (ZODB) es una base de datos orientada a objetos que ofrece una forma casi transparente de persistencia para aplicaciones escritas en Python y está diseñada para tener poco impacto en el código de la aplicación.

ZODB soporta transacciones, versiones de objetos y puede ser conectada a otros backends a través de Zope Enterprise Objects (ZEO), permitiendo incluso la creación de aplicaciones distribuidas a través de múltiples máquinas en red.

ZODB es un componente integral de Zope 29, que es un servidor para aplicaciones desarrolladas en Python, ampliamente utilizado en Sistemas de Gestión de Contenidos (CMS).

Página de formato en: http://www.json.org/.

Página de formato en: http://yaml.org/.

Página de formato en: http://www.w3.org/XML/.

La documentación y los paquetes de instalación de Zope y los productos conexos en http://www.zope.org/.

Componentes de ZODB:

Base de datos : Permite a la aplicación hacer conexiones (interfaces de acceso a objetos).

Transacción : interfaz que permite hacer cambios permanentes.

Persistencia : Proporciona la clase de base Persistente.

Almacenamiento : Administra la representación persistente del disco.

ZEO : Compartir objetos entre diferentes procesos y máquinas.

Ejemplo de uso de ZODB:

from ZODB import FileStorage, DB

import transaction

Item went back to what it was before the transaction print root ['singer'] # Kate Bush

La ZODB tiene algunas limitaciones que deben tenerse en cuenta durante el diseño de la aplicación:

Los objetos deben ser "serializables" para ser almacenados.

Los objetos cambiantes requieren un cuidado especial.

Los objetos "serializables" son aquellos que pueden ser convertidos y recuperados por Pickle. Entre los objetos que no pueden ser procesados por Pickle están los implementados en módulos escritos en C, por ejemplo.

YAML

YAML es un formato de serialización de datos de texto que representa los datos como combinaciones de listas, diccionarios y valores escalares. Su principal característica es que es legible para el ser humano.

El proyecto YAML fue fuertemente influenciado por la sintaxis de Python y otros lenguajes dinámicos. Entre otras estructuras, la Especificación 30 de YAML establece que:

Los bloques están marcados por una hendidura.

Las listas están entre paréntesis o indicadas por guiones.

Las teclas del diccionario van seguidas de dos puntos.

Las listas pueden representarse de la siguiente manera:

Azul
Blanco
Rojo
O:
[azul, blanco, rojo]
Los diccionarios se representan como:
Color : Blanco
Nombre: Bandido
Raza : Labrador
PyYAML 31 es un módulo de rutinas para generar y procesar YAML en Python.

MVC

El modelo-vista-controlador (MVC) es una arquitectura de software que divide la aplicación en tres partes distintas: el modelo de datos de la aplicación, la interfaz de usuario y la lógica de control.

El objetivo es acoplarse libremente entre las tres partes de modo que un cambio en una parte tenga poco (o ningún) impacto en las otras partes.

MVC: Controlador de la vista del modelo

Recupera los datos y el controlador recibe y reacciona a los eventos del usuario.

La creación de la aplicación depende de la definición de tres componentes:

Modelo (model): encapsula los datos de la aplicación y la lógica del dominio.

Visión (vista): recupera los datos del modelo y presenta al usuario.

Controlador (controller): recibe y reacciona a posibles eventos como las interacciones del usuario y solicita cambios en el modelo y la visión.

Aunque la arquitectura no determina formalmente la presencia de un componente de persistencia, se implica que es parte del componente del modelo.

El uso más común del modelo MVC es en las aplicaciones web basadas en bases de datos que implementan las operaciones básicas llamadas CRUD (Create, Read, Update, and Delete).

Existen varios marcos de trabajo para aumentar la productividad de la creación de aplicaciones siguiendo el MVC, con características como:

Scripts que automatizan las tareas de desarrollo más comunes.

Generación automática de código.

Uso de ORM.

Uso de CSS 49 (Hojas de Estilo en Cascada).

Uso de Javascript asíncrono y XML (AJAX).

Plantillas de aplicación.

Uso de insight para recoger información sobre estructuras de datos y generar formularios con campos con características correspondientes.

Varias opciones están preconfiguradas con valores por defecto adecuados para la mayoría de las aplicaciones.

Interfaz Grafica

La Interfaz Gráfica de Usuario (GUI) se ha hecho popular en el entorno de los ordenadores de sobremesa debido a su facilidad de uso y su productividad. Ahora hay muchas bibliotecas disponibles para construir aplicaciones GUI como GTK +, QT, TK, y wxWidgets.

Arquitectura

Las interfaces gráficas a menudo utilizan la metáfora del escritorio, un espacio en dos dimensiones, que está ocupado por ventanas rectangulares que representan aplicaciones, propiedades o documentos.

Window

Las ventanas pueden contener varios tipos de controles (objetos utilizados para interactuar con el usuario o presentar información) y contenedores (objetos que sirven como depósitos para colecciones de otros objetos).

La mayoría de las veces, la interfaz gráfica espera a que ocurran eventos y responde en consecuencia. Los eventos pueden ser el resultado de la interacción del usuario, como los clics del ratón y el arrastre o la escritura, o los eventos del sistema, como el reloj de la máquina. La reacción a cualquier evento se define mediante funciones de devolución de llamada (funciones que se pasan como argumentos a otras rutinas).

Controles más utilizados:

Etiqueta (label): rectángulo que muestra el texto.

Caja de texto : área de edición de texto.

Botón (button): área que puede ser activada interactivamente.

Botón de radio (radio button): un tipo especial de botón, con el que se forman grupos donde sólo se puede apretar uno a la vez.

Botón de comprobación (botón de comprobación): un botón que puede ser comprobado y deshecho.

Barras de desplazamiento : Deslizadores horizontales o verticales utilizados para los rangos de valores numéricos.

Perilla (botón de giro): caja de texto con dos botones con flechas junto a ese decremento e incremento del número en la caja.

Barra de estado (barra de estado): barra para mostrar mensajes, generalmente en la parte inferior de la ventana.

Imagen (imagen): área para la visualización de imágenes.

Los controles pueden tener aceleradores (teclas de acceso directo) asociados.

Contenedores de uso común:

Barra de menú (barra de menú): sistema de menús, normalmente en la parte superior de la ventana.

Fijo (fijo): los objetos permanecen fijos en las mismas posiciones.

Tabla (Tabla): una colección de compartimentos para asegurar los objetos, dispuestos en filas y columnas.

Caja horizontal (caja horizontal): Similar a la tabla, pero sólo una línea.

Case vertical (vertical box): Similar to the table, but only one column.

Notebook (laptop): varias áreas que pueden ser vistas una a la vez a través de pestañas cuando se seleccionan, generalmente en la parte superior.

Barra de herramientas: área con botones para el acceso rápido a las funciones clave de la aplicación.

Para tratar los eventos de tiempo, las interfaces gráficas implementan una característica llamada temporizador (timer) que evoca la función de devolución de llamada después de un cierto tiempo establecido.

PyGTK

GTK + 60 (GIMP Toolkit) es una biblioteca de código abierto escrita en lenguaje C. Diseñada originalmente para ser usada por GIMP 61 , es compatible con las plataformas más ricas en características de hoy en día, incluyendo un constructor de interfaces llamado Glade.

El sitio web del proyecto está en: http://www.gtk.org/ . y los binarios de Windows están disponibles en: http://gladewin32.sourceforge.net/ . La versión para desarrolladores instala Glade.

Dirección oficial del proyecto: http://www.gimp.org/ .

Documentación y fuentes en: http://www.gnome.org/ .

Sistemas UNIX. GTK+ puede ser usado en Python a través del paquete PyGTK 63. Los puertos de la biblioteca para Windows se pueden encontrar en :

PyGTK: http://ftp.gnome.org/pub/gnome/binaries/win32/pygtk/ .

PyGObject: http://ftp.gnome.org/pub/gnome/binaries/win32/pygobject/

PyCairo: http://ftp.gnome.org/pub/gnome/binaries/win32/pycairo/

Aunque es posible crear interfaces completamente usando código, es más productivo construir la interfaz en el software apropiado. Glade genera archivos XML con la extensión ".glade" que pueden ser leídos por los programas que utilizan GTK+, automatizando el proceso de creación de interfaces gráficas.

Hoja de ruta básica para la construcción de una interfaz:

No hay ningún Glade:

Crea una ventana usando cualquiera de las plantillas disponibles en "Niveles superiores".

Crear contenedores para almacenar los controles.

Crear controles.

Crear los manipuladores para las señales requeridas.

Guardar el archivo con la extensión ".glade".

En Python:

Importar los paquetes requeridos.

Usar GTK para interpretar el archivo XML de Glade.

Crear rutinas para ser usadas como funciones de devolución de llamada.

Asocie las rutinas con los manejadores de correspondencia que se crearon en Glade usando el método signal_autoconnect () .

Habilitar el bucle para procesar eventos con gtk.main () .

Ejemplo (reloj):

No hay Glade:

Haga clic en "Ventana" en "Niveles superiores".

En las propiedades de la ventana:

- Cambie el "Nombre" por "principal" en el "General".

La página web de PyGTK es http://www.pygtk.org/ .

Cambie "Redimensionar" por "Sí".

Cambiar "Posición de la ventana" a "Centro".

Cambiar "Visible" por "Sí" en "Común".

Cambiar el manejador a "on_main_destroy" de la señal "destroy" de "GtkObject" en "Señales".

Haz clic en "Caja vertical" en "Contenedores", luego haz clic en el interior de la ventana y elige el número de elementos igual a 3.

Haga clic en "Barra de menú" en "Contenedores", luego haga clic dentro del espacio vacío superior y elimine los elementos "Editar" y "Ver".

Haga clic en "Barra de estado" en "Control y visualización" y luego haga clic dentro del espacio vacío inferior.

Cambie el nombre de la barra de estado a "sts_data" en "General".

Haga clic en "Etiqueta" en "Control y visualización" y luego haga clic dentro del espacio vacío central.

En las propiedades de la etiqueta, cambie "Name" a "lbl_hora" y "Label" a vacío en "General", "Width Request" a "300" y "Height Request" a "150" en "Common".

En el Inspector (lista de árbol con todos los elementos), eliminar:

“Imageenuitem1”.

“Imageenuitem2”.

“Imageenuitem3”.

“Imageenuitem4”.

“Separatormenuitem1”.

En el Inspector:

encontrar "imageenuitem5" y cambiar el manejador en "Señales" de la señal "activar" a "on_imagemenuitem5_activate" de "GtkMenuItem".

encontrar "imageenuitem10" y cambiar el handler en "Señales" de la señal "activate" a "on_imagemenuitem10_activate" de "GtkMenuItem".

Guarda el archivo como "clock.glade".

wxPython

El paquete wxPython es básicamente un envoltorio para el conjunto de herramientas (conjunto de herramientas y librerías) wxWidgets, desarrollado en C ++. Características principales:

Multiplataforma.

Aspecto nativo, es decir, consistente con el entorno en el que se ejecuta.

Una gran colección de componentes listos para usar.

Una comunidad muy activa.

La forma general de funcionamiento es similar a GTK +: el marco controla la interacción del usuario a través de un bucle que detecta los eventos y activa las rutinas correspondientes.

La forma más común de implementar una interfaz gráfica a través de wxPython es:

Importar el paquete wx .

Crear una clase de ventana a través de la herencia. En la inicialización de la clase se pueden definir los controles y contenedores que forman parte de la ventana y las asociaciones entre eventos con las funciones de devolución de llamada correspondientes , que suelen ser métodos de la propia clase .

Crear un objeto "aplicación" utilizando la clase wxPython App .

Crear un objeto de la clase window .

Iniciar el bucle de manejo de eventos de la aplicación .

Otras funcionalidades asociadas a la interfaz gráfica se pueden obtener utilizando otros módulos, como pySystray 69 , que implementa una funcionalidad que permite a la aplicación utilizar la bandeja del sistema en Windows.

CG

Gráficos por Computadora (CG) es el área de la Ciencia de la Computación que estudia la generación, representación y manipulación de contenido visual en sistemas de computación y tiene aplicaciones en varias áreas del conocimiento humano.

Las simulaciones, por ejemplo, son sistemas que emplean cálculos matemáticos para imitar uno o más aspectos de un fenómeno o proceso que existe en el mundo real. Las simulaciones permiten comprender mejor cómo funciona el experimento real y comprobar escenarios alternativos con otras condiciones.

En el caso de los juegos, que son en realidad una forma de simulación interactiva que utiliza medios visuales para aumentar la sensación de realismo, lo que se conoce como inmersión, y por lo tanto enriquece la experiencia del jugador.

Otra aplicación es la visualización, como dice un viejo dicho popular: "una imagen vale más que mil palabras", y esto es aún más cierto cuando se trata de interpretar grandes cantidades de datos, como es el caso de muchas actividades científicas, médicas y de ingeniería.

Áreas como la geografía, la cartografía y la geología requieren Sistemas de Información Geográfica (SIG), que representan topologías y datos asociados como la altura, la humedad y otros.

La ingeniería y las actividades conexas utilizan herramientas de diseño asistido por computadora (CAD) para facilitar la creación de dibujos técnicos de componentes o piezas de maquinaria.

Además, varias formas de arte se benefician de la CG, como el cine, especialmente para la creación de efectos especiales. El CG también ha permitido el surgimiento de nuevas formas de arte que utilizan un entorno digital como medio, como la animación 3D.

Arreglos vs Vectores

Es muy común representar la información visual de forma bidimensional (2D), ya sea en fotos, gráficos impresos o en una pantalla LCD. Existen dos formas de representación de imágenes bidimensionales ampliamente utilizadas, cada una con sus ventajas y desventajas.

La primera matriz también se conoce como bits de mapa (bitmap) o raster , en la que la imagen se representa como una matriz bidimensional de puntos con información de color, llamados elementos de imagen (picture element , comúnmente abreviado como pixel). Esta forma requiere la manipulación y el almacenamiento de sofisticados algoritmos, debido al volumen de los datos y a la complejidad de las operaciones, como la interpolación de valores durante el cambio de tamaño, por ejemplo.

La segunda forma de representación es la de las imágenes vectoriales, que se describen a través de entidades matemáticas que conforman la geometría de la imagen (líneas, polígonos, texto y otros). Esta forma es menos exigente en cuanto a recursos computacionales y no presenta problemas asociados con el escalamiento, pero no permite muchas operaciones que el mapa de bits permite.

Entre otras formas de representación, es interesante destacar los fractales, en los que las imágenes se generan a través de algoritmos que se aplican de forma similar.

Estas formas de representación han dado lugar a la aparición de diversos formatos de archivo para almacenar imágenes, incluso abiertas, como Portable Network Graphics (PNG), que admite incluso imágenes raster transparentes , y Scalable Vectorial Graphics (SVG) para imágenes vectoriales. , mapas de bits, e incluso animaciones. Ambos están aprobados por el Consorcio de la World Wide Web (W3C).

Hay ahora varias bibliotecas orientadas a CG disponibles para Python, que están en una etapa avanzada de madurez.

Procesamiento de Imágenes

La Biblioteca de Imágenes de Pitón (PIL) es una biblioteca de procesamiento de matrices para Pitón.

PIL tiene módulos que implementan:

Herramientas para recortar, redimensionar y fusionar imágenes.

Algoritmos de conversión, que soportan varios formatos.

Filtros, como suavizar, desenfocar y detectar bordes.

Ajustes, incluyendo brillo y contraste.

Operaciones con paletas de color.

Dibujos 2D simples.

Rutinas para el procesamiento de imágenes: ecualización, autocontraste, deformación, inversión y otras.

Ejemplo de procesamiento de imágenes:

- * - coding: latin-1 - * -

"" "

Crea miniaturas suavizadas para cada JPEG de la carpeta actual.

"" "

import glob

Importación de la PIL Módulo principal de la imagen

Import ImageFilter filter module

SVG

Los archivos SVG pueden almacenar varios tipos de información vectorial, incluidos los polígonos básicos, que están representados por líneas que delimitan un área cerrada, como rectángulos, elipses y otras formas simples. Además, también admite caminos (paths), que son figuras con relleno o no, compuestas por líneas y/o curvas definidas por puntos, que se codifican mediante comandos de un carácter ("L" significa "Line To" por ejemplo) y un par de coordenadas X e Y, lo que genera un código muy compacto.

El texto Unicode puede incluirse en un archivo SVG, con efectos visuales, y la especificación incluye el manejo de texto bidireccional, vertical y siguiendo trayectorias curvas. El texto se puede formatear con fuentes externas de texto, pero para aliviar el problema de que el texto no se

muestre correctamente en los diferentes sistemas, hay una fuente interna que siempre está disponible.

Las figuras geométricas, los trayectos y el texto pueden utilizarse como contornos, por dentro o por fuera, que pueden utilizar tres tipos de relleno:

Colores sólidos, que pueden ser opacos o transparentes.

Gradientes, que pueden ser lineales o radiales.

Patrones, que son imágenes de mapa de bits o vectoriales que se repiten en todo el objeto.

Se pueden animar tantos gradientes como patrones.

SVG también permite al autor incluir metadatos con información sobre la imagen, como el título, la descripción y otros, para facilitar la catalogación, la indexación y la recuperación de archivos.

Todos los componentes de un archivo SVG pueden ser leídos y cambiados usando scripts de la misma manera que el HTML, por defecto en el lenguaje ECMAScript . La especificación también proporciona el manejo de eventos de ratón y teclado , que, junto con los hipervínculos, permite añadir interactividad a los gráficos.

El formato también soporta la animación a través de ECMAScript , que puede transformar los elementos de la imagen y cronometrar el movimiento. Esto también se puede hacer a través de los recursos propios de SVG utilizando etiquetas .

Para el SVG, los filtros son conjuntos de operaciones gráficas que se aplican a un gráfico vectorial determinado para producir una imagen matriz con el resultado. Estas operaciones gráficas se denominan primitivas de filtro, que suelen realizar una forma de procesamiento de la imagen, como el efecto de Desenfoque Gaussiano , y así generan un mapa de bits con transparencia (estándar RGBA) como salida, que se regenera si es necesario. El resultado de una primitiva puede utilizarse como entrada de otra primitiva, permitiendo la concatenación de varias para generar el efecto deseado.

SVGFig

Los archivos SVG pueden manipularse a través de bibliotecas XML, como ElementTree, pero es más productivo utilizar componentes ya diseñados para este fin. SVGFig es un módulo SVG listo para ser utilizado. El módulo permite utilizar directamente tanto primitivas de dibujo SVG como rutinas propietarias de alto nivel.

SVGFig tiene varias primitivas de dibujo implementadas en forma de funciones, entre las que se incluyen rutas (Path ()), líneas (Line ()) y texto (Text ()).

Imágenes Tridimensionales

Los formatos matriciales y vectoriales representan imágenes bidimensionales en la computadora que son adecuadas para la mayoría de las aplicaciones. Sin embargo, están limitados en muchos aspectos, especialmente para las simulaciones, ya que el mundo en el que vivimos tiene tres dimensiones (3D).

Una escena 3D está compuesta por objetos que representan sólidos, fuentes de luz y cámaras. Los objetos sólidos suelen representarse con mallas, que son un conjunto de puntos (vértices). Estos tienen coordenadas x, y, y z. Los puntos están interconectados por líneas (bordes) que forman las superficies (caras) de los objetos. Los conjuntos de líneas que representan las mallas se denominan "wireframes".

Objetos

Los objetos pueden utilizar uno o más materiales, y éstos pueden tener varias características, como el color, la transparencia y el sombreado, que es la forma en que el material responde a la iluminación de la escena. Además, el material puede tener una o más texturas asociadas.

Las texturas están compuestas por imágenes bidimensionales que pueden ser utilizadas en materiales aplicados a las superficies de los objetos, cambiando varias propiedades como el reflejo de la superficie, la transparencia y la protuberancia.

En una escena tridimensional, los objetos pueden modificarse mediante transformaciones como la traslación (pasar de una posición a otra), la rotación (girar alrededor de un eje) y el cambio de tamaño (redimensionar una o más dimensiones).

Wireframe Con imagenes solidas

Para renderizar, es decir, generar la imagen final. Necesitas hacer una serie de cálculos complejos para aplicar la iluminación y la perspectiva a los objetos de la escena. Entre los algoritmos utilizados para la renderización, uno de los más conocidos es el llamado raytrace , en el que

se calculan los rayos de luz desde la cámara hasta las fuentes de luz. Esto evita cálculos innecesarios de rayos que no llegan a la cámara.

Uno de los usos más populares de la tecnología 3D es en las animaciones. La técnica de animación 3D más común se llama "keyframe". En él, el objeto a animar se posiciona en diferentes lugares en los momentos clave de la animación, y el software se encarga de calcular los cuadros intermedios.

Muchas aplicaciones 3D utilizan bibliotecas que implementan la especificación OpenGL (Open Graphics Library), que define una API independiente de la plataforma y el lenguaje para manipular los gráficos 3D, lo que permite un renderizado en tiempo real acelerado por hardware. Su característica más llamativa es el rendimiento.

VPython

VPython es un paquete que permite crear y animar modelos tridimensionales simples. Su objetivo es facilitar la rápida creación de simulaciones y prototipos que no requieren soluciones complejas.

VPython proporciona iluminación, control de cámara y manejo de eventos de ratón (rotación y zoom) de forma automática. Los objetos pueden ser creados de forma interactiva en el intérprete, que la ventana 3D de VPython se actualiza en consecuencia.

VPython tiene varias limitaciones. No proporciona formas de crear y/o manipular materiales o texturas más complejas, ni formas avanzadas de iluminación o detección de colisiones. Para escenas más sofisticadas, hay otras soluciones, como Python Ogre 75 y Blender, que es una aplicación de modelado 3D que utiliza Python como lenguaje de scripting.

Py OpenGL

Las bibliotecas OpenGL 76 implementan una API de manipulación de imágenes 3D de bajo nivel, que permite el acceso a los recursos disponibles en el hardware de vídeo , y también hacen que el código sea independiente de la plataforma, ya que el software emula características no disponibles en el equipo. Estas características incluyen primitivas (líneas y polígonos), mapeo de texturas, operaciones de transformación e iluminación.

OpenGL funciona en un contexto, cuyo estado se modifica a través de las funciones definidas en la especificación. Este estado se mantiene hasta que se vuelve a cambiar.

Además de la biblioteca principal, la OpenGL Utility Library (GLU) es una biblioteca de alto nivel, mientras que el OpenGL Utility Toolkit (GLUT) define rutinas independientes de la plataforma para la gestión de ventanas, entradas y contextos.

La GLUT está orientada a eventos, a los que se pueden asociar funciones de devolución de llamada , que ejecutan llamadas OpenGL. La biblioteca tiene una rutina que monitoriza los eventos y evoca las funciones cuando es necesario.

Py OpenGL es un paquete que permite a los programas Python utilizar las bibliotecas OpenGL, GLU y GLUT.

Borra la matriz de proyección glLoadIdentity ()

Calcula el aspecto de la perspectiva gluPerspective (45. , flotador (x) / flotador (y), 0.1 , 100.0)

Selecciona la matriz de visualización glMatrixMode (GL_MODELVIEW)

```
def draw ():
"" "
Funcion que Dibuja Objetos
"" "
Global air , dr
# Ventana despejada y amortiguador de profundidad
glClear ( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT )
Clears the glLoadIdentity () view matrix
Mover el objeto (traducción)
Parámetros: x, y and z (cambio) glTranslatef (- 0.5 , - 0.5 , - 4. )
Rotación (en grados)
Parámetros: grados, x, y and z (ejes) glRotatef ( ar , 1.0 , 1.0 , 1.0 )
Escalas
Parámetros: x, y and z (tamaño)
glScalef ( air / 1000 , air / 1000 , air / 1000 )
for i in xrange ( 0 , 360 , 10 ):
Rotation of glRotatef Object Faces ( 10 , 1.0 , 1.0 , 1.0 )
Starts creating a glBegin rectangular face ( GL_QUADS )
```

Defines the color to use to draw (R, G, B) glColor3f (.7 , .5 , .1)
Create a face vertex
glVertex3f (0. , 0. , 0.)
glColor3f (.7 , .3 , .1)
glVertex3f (1. , 0. , 0.)
glColor3f (.5 , .1 , .1)
glVertex3f (1. , 1. , 0.)
glColor3f (.7 , .3 , .1)
glVertex3f (0. , 1. , 0.)
Ends the face glEnd ()
Inverts the variation if ar > 1000 : dr = - 1 if ar < 1 : dr = 1
air = air + dr
Intercambiar el buffer, mostrando lo que se acaba de usar glutSwapBuffers ()
def keyboard (* args):
"" "
Función de llamada para manejar los eventos del teclado
"" "
Prueba si la tecla ESC fue presionada if args [0] == '\ 33' :
raise SystemExit
if __name__ == '__main__' :
Inicializa GLUT glutInit (argv)
Seleccionar el modo de display
glutInitDisplayMode (GLUT_RGBA | GLUT_DOUBLE | GLUT_DEPTH)
Establece la resolución de la ventana de GLUT de 640 x 480 glutInitWindowSize (640 , 480)
Crear una GLUT window
window = glutCreateWindow ('Transformations')
# Configurar la función de llamada que se basa en la ventana actual

glutDisplayFunc (draw)

Para ver la pantalla completa #glutFullScreen ()
Registra la función para manejar el redibujo de la ventana cuando no hay nada que hacer
glutIdleFunc (draw)
Registra la función para redibujar la ventana cuando se redimensiona
glutReshapeFunc (resize)
Función de registro para manejar los eventos del teclado glutKeyboardFunc (keyboard)
Inicialización de Ventana
Limpia la imagen (black background)
glClearColor (0. , 0. , 0. , 0.)
Limpia el buffer profundo glClearDepth (1.)
Configura el tipo de analisis profundo glDepthFunc (GL_LESS)
Permite el análisis profundo
glEnable (GL_DEPTH_TEST)
configura el sombreado suave glShadeModel (GL_SMOOTH)
Selecciona la matriz de proyección glMatrixMode (GL_PROJECTION) glLoadIdentity ()
gluPerspective (45 , 640 / 480 , .1 , 100.)
Selecciona la matriz de visualización glMatrixMode (GL_MODELVIEW)
Inicie el evento GLUT glutMainLoop () ()
Salir de la ventana:

Imágenes Tridimensionales 267

OpenGL puede integrarse con conjuntos de herramientas gráficas , como wxWidgets y Qt , en lugar de usar GLUT, y así incorporarse a las aplicaciones GUI convencionales.

Una de las principales referencias sobre OpenGL es el libro "Guía de Programación de OpenGL", también conocido como "Libro Rojo" 78 .

La versión online está disponible en http://www.glprogramming.com/red/ .

Procesamiento distribuido

Generalmente la solución a los problemas que requieren mucha potencia de cálculo es el uso de máquinas más potentes, pero esta solución es limitada en cuanto a la escalabilidad. Una alternativa es dividir los procesos de aplicación entre múltiples máquinas que se comunican a través de una red, formando un clúster o una red.

La diferencia básica entre el cluster y la grid es que el primero se basa en la premisa de que el diseño debe ser un entorno controlado, homogéneo y predecible, mientras que el segundo es generalmente heterogéneo, incontrolado e impredecible. Un cluster

un entorno diseñado específicamente para el procesamiento distribuido, con máquinas dedicadas en un solo lugar. Una red se caracteriza por el uso de estaciones de trabajo que pueden estar en cualquier lugar.

El modelo computacional tiene por objeto utilizar los procesadores y la memoria del equipo involucrado para obtener más potencia de cálculo. La aplicación suele utilizar un sistema de metaplanificador, que programa las tareas que han de ser procesadas por los nodos (máquinas que componen el modelo), por lo que la operación tiende a ser continua, con una interacción reducida con los usuarios. Un ejemplo conocido es SETI @ home 79 .

El recurso de clúster se utiliza para almacenar información en un grupo de computadoras, tanto para aumentar el rendimiento de la recuperación de datos como para ampliar la capacidad de almacenamiento. Esta plantilla puede utilizarse para proporcionar una infraestructura para las aplicaciones o para responder a las solicitudes formuladas de manera interactiva por los usuarios. Entre los servicios que pueden funcionar de esta manera se encuentran los sistemas de gestión de bases de datos (DBMS), como el MySQL Cluster 80 .

El modelo híbrido es una aplicación diseñada específicamente para trabajar con

Página del proyecto en: http://setiathome.berkeley.edu/ .

Dirección de Internet: http://www.mysql.com/products/database/cluster/ .

en múltiples máquinas a la vez. En lugar de proporcionar recursos directamente, la aplicación utiliza el equipo para apoyar su propia funcionalidad. Como resultado, la infraestructura es utilizada de manera casi transparente por los usuarios que utilizan la aplicación de manera interactiva. Todos los nodos ejecutan la aplicación y pueden funcionar como servidores y clientes. El ejemplo más común de arquitectura híbrida son los sistemas de intercambio de archivos que utilizan la comunicación entre pares (P2P).

Independientemente del modelo utilizado. Los sistemas distribuidos deben cumplir cuatro requisitos básicos:

Comunicación: Las máquinas involucradas deben comunicarse de tal manera que permitan el intercambio de información entre ellas.

Metadatos: El procesamiento de los datos debe mantenerse adecuadamente.

Control: Los procesos deben ser administrados y controlados.

Seguridad: La confidencialidad, la integridad y la disponibilidad deben ser protegidas.

Existen varias tecnologías enfocadas al desarrollo de aplicaciones distribuidas, tales como: XML-RPC 81 , Web Services, objetos distribuidos, MPI, y otros.

Especificaciones en http://www.xmlrpc.com/ .

Objetos distribuidos

La premisa básica de la tecnología de objetos distribuidos es hacer que los objetos estén disponibles para que sus métodos puedan ser evocados remotamente desde otras máquinas o incluso otros procesos en la misma máquina utilizando para ello la pila de protocolos de red TCP / IP.

Existen varias soluciones para estos casos, pero el uso de objetos distribuidos ofrece varias ventajas sobre otras soluciones que implementan una funcionalidad similar, como el protocolo XML-RPC:

Simplicidad de implementación.

Oculta las capas de comunicación.

Soporte de estructuras de datos nativas (siempre que sean serializables).

Buen rendimiento.

Madurez de la solución.

PYthon Remote Objects (PYRO 82) es un marco de trabajo para aplicaciones distribuidas.

82 Documentación y fuentes disponibles en: http://pyro.sourceforge.net/ .

que le permite publicar objetos a través de TCP / IP. En la máquina del servidor, PYRO publica el objeto, cuidando detalles como el protocolo, el control de sesión, la autenticación, el control de concurrencia y otros.

La comprensión de la lista es más eficiente que el empate tradicional.

Otra forma de mejorar el rendimiento de una aplicación es usando Psyco , que es una especie de Just In Compiler (JIT). En tiempo de ejecución, trata de optimizar el código de la aplicación, por lo que el módulo debe ser importado antes que el código a optimizar (el comienzo del módulo principal de la aplicación es un

85 El módulo cProfile no es adecuado para evaluar fragmentos cortos de código. El tiempo que modula es el más adecuado porque ejecuta el código varias veces durante la evaluación.

lugar apropiado).

Ejemplo (con el último fragmento de código evaluado en el ejemplo anterior):

importar psyco

Trata de optimizar todo lo que sea psicológico.

importar timeit

Lista de cuadrados del 1 al 1000

cod = '[x ** 2 for x in xrange (1, 1001)]' print timeit . Timer (cod). timeit ()

Salida:

127.678481102

El código corrió más del doble de rápido que antes. Esto requirió sólo añadir dos líneas de código.

Sin embargo, Psyco debe ser usado con cierta precaución, ya que en algunos casos, puede no ser capaz de optimizar o incluso empeorar el rendimiento 86 . Las funciones de mapa () y filtro () deben ser evitadas, y los módulos escritos en C, como re (expresiones regulares), deben ser marcados con la función can not compile () para que Psyco los ignore. El módulo proporciona formas de optimizar sólo ciertas partes del código de la aplicación, como la función de perfil () , que sólo optimiza las partes más pesadas de la aplicación y una función de registro () que analiza la aplicación para evitar estas situaciones.

Algunos consejos de optimización:

Mantenga el código simple.

Sólo optimice el código cuando el rendimiento de la aplicación sea realmente crítico.

Hay algunas funciones en las que Psyco tiene el efecto de ralentizar el programa porque el propio Psyco también consume CPU. Además, Psyco también hace que la aplicación consuma más memoria.

Usar herramientas para identificar los cuellos de botella en el código.

Evitar las funciones recursivas.

Utilizar funciones en el idioma nativo. Las listas y diccionarios en Python están muy optimizados.

Utilice Comprensiones de listas en lugar de bucles para procesar listas utilizando expresiones simples.

Evite las funciones dentro de los bucles. Las funciones pueden recibir y devolver listas.

Use generadores en lugar de funciones para grandes cadenas de datos.

Empaquetado y distribución

Suele ser mucho más sencillo distribuir aplicaciones en forma binaria, donde simplemente se ejecuta un ejecutable para lanzar la aplicación que instalar todas las dependencias necesarias en cada máquina en la que se desea ejecutar la aplicación.

Existen programas que permiten generar ejecutables a partir de un programa hecho en Python, como Py2exe 87 y cx_Freeze 88 .

Py2exe funciona sólo en la plataforma Windows, pero tiene muchas características y puede generar ejecutables con una interfaz de texto, gráficos, servicios (programas que se ejecutan sin la intervención del usuario, de manera similar a los demonios de los sistemas UNIX) y servidores COM (arquitectura de componentes de Microsoft).

87 La documentación, las fuentes y los binarios para la instalación se pueden encontrar en

http://www.py2exe.org/ .

88 La documentación de instalación multiplataforma, las fuentes y los binarios se pueden encontrar en:
http://starship.python.net/crew/atuining/cx_Freeze/ .

284 Empaquetado y distribución

Cx_Freeze es portátil y puede funcionar en entornos UNIX, pero es mucho menos versátil que Py2exe.

Para usar Py2exe, es necesario crear un script , comúnmente llamado "setup.py", que le dice a Py2exe lo que se necesita para generar el ejecutable.

Ejemplo de “setup.py”:

```
- * - coding: latin1 - * -
    "" "
    Py2exe usage example
    "" "
    from distutils . core import setup import py2exe
    setup ( name = 'SIM - Interactive Music System' , service = [
'simservice' ],
    console = [ 'sim.py' , 'simimport.py' ],
    windows = [ 'simgtk.py' ],
    options = { 'py2exe' : {
    'optimize' : 2 ,
    'includes' : [ 'atk' , 'gobject' , 'gtk' , 'gtk.glade' , 'pango' , 'cairo' ,
'pangocairo' ]
    }},
    data_files = [( '' , [ 'Janela.glade' , 'sim.ico' ])], description = 'First
Version ...' , version = '1.0' )
```

En el ejemplo, tenemos un sistema que consiste en dos utilidades de línea de comando, una aplicación de interfaz gráfica y un servicio. La aplicación de interfaz gráfica depende de GTK + para funcionar y fue desarrollada usando Glade.

Entre los parámetros de Py2exe, los más comunes son:

nombre : nombre de la aplicación.

service : lista de servicios.

console : lista de programas con una interfaz de texto.

windows : lista de programas con una interfaz gráfica.

options ['py2exe'] : Diccionario con opciones que cambian el comportamiento de Py2exe:

- optimize : 0 (optimización desactivada, bytecode por defecto), 1 (optimización

habilitado, equivale al parámetro "-O" del intérprete) o 2 (optimización con la eliminación de Doc Strings habilitada, equivale al parámetro "-OO" del intérprete).

incluye : lista de módulos que se incluirán como dependencias. Py2exe suele detectar las dependencias sin tener que utilizar esta opción.

archivos_de_datos : Otros archivos que forman parte de la aplicación, como imágenes, iconos y archivos de configuración.

description : comentario.

version : una versión de la aplicación como cadena .

Para generar el ejecutable, el comando es:

```
python setup.py py2exe
```

Py2exe creará dos carpetas:

build : archivos temporales.

dist : archivos para distribución.

Entre los archivos para la distribución, "w9xpopen.exe" sólo es necesario para las versiones antiguas de Windows (95 y 98) y puede ser

eliminado sin problemas en las versiones más recientes.

A través de la línea de comandos, también se pueden pasar algunas opciones interesantes, como el parámetro "-b1", para generar menos archivos para su distribución.

Conclusión

Gracias por leer Python para el análisis de datos, esperamos que haya sido capaz de proporcionarle todas las herramientas que necesita para alcanzar sus objetivos, sean cuales sean.

Este libro trata de módulos y paquetes, destacando algunos de los más relevantes que están presentes en la biblioteca del lenguaje estándar, la instalación de la biblioteca de terceros, las excepciones y la introspección.

En la actualidad hay mucho código heredado desarrollado en varios lenguajes que Python puede aprovechar a través de varias formas de integración.

Una forma genérica de hacerlo es generar una biblioteca compartida a través del compilador del otro lenguaje y hacer llamadas a las funciones que se definen en la biblioteca. Dado que la implementación original de Python utiliza el lenguaje C, es posible integrar Python y C de dos formas:

- Python -> C (Python hace llamadas a un módulo compilado en C).
- C -> Python (C evoca el intérprete de Python en modo embebido).

Esperamos que hayan disfrutado de la lectura de este libro y que hayan aprendido mucho de él.