Python 3 para impacientes

"Simple es mejor que complejo" (Tim Peters)

Python

Última entrada:
Data Classes: Clases de Datos

Indice

Buscar

Buscar

Python para impacientes

Python
IPython
EasyGUI
Tkinter
JupyterLab
Numpy

Anexos

Guía urgente de MySQL
Guía rápida de SQLite3

Entradas + populares

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

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

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

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

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

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

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

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

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

Threading: programación con hilos (I)
En programación, la técnica que permite que una aplicación ejecute

- Tuplas
- Operaciones con cadenas y listas
- Operaciones con cadenas y tuplas
- Diccionarios o matrices asociativas
- Operaciones con diccionarios
- Recorrer secuencias y diccionarios con for...in
- Recorrer rangos con for...in range()
- Operadores para secuencias: in, not in, is, is not
- Conjuntos: set
- Operaciones con conjuntos

Tipos de cadenas: Unicode, Byte y Bytearray

- Un poco de historia
- Cadenas Unicode, Byte y Bytearray
- Declarar y convertir cadenas
- Codificación de archivos de texto

Entrada Estándar: input()

- Introducir datos de distinto tipo
- Introducir datos con captura de errores (excepciones)
- Introducir datos con captura de errores en bucle
- Introducir fechas y horas con captura de errores

Salida Estándar: print()

Las cadenas f

- Formateo basado en %, en str.format() y string.Template()
- Las cadenas de formato f

Programas con estilo en Python

- Recomendaciones de PEP 8
- Convenciones de nombres
- Diseño
- Líneas de código
- Codificación
- Líneas en blanco
- Sangría
- Comentarios
- Espacios en blanco
- Otras recomendaciones
- Herramientas para revisar y corregir

Dar color a las salidas en la consola

- Aplicar formatos con códigos ANSI
- El módulo Colorama
- Aplicar formatos con Colorama
- Desplazar el cursor antes de dar formato

Excepciones

- Excepciones
- Excepciones a medida
- De obligado cumplimiento: assert

Funciones

- Funciones con un número fijo de parámetros
- Funciones con un número variable de parámetros
- Funciones con parámetros con valores por defecto
- Funciones con todos los parámetros con valores por defecto
- Funciones con parámetros que contienen diccionarios
- Funciones que devuelven más de un valor
- Funciones sin return
- Funciones generadas a partir de otras

Funciones que imponen nombrar o no parámetros

Iteradores y generadores

- Iteradores
- La función iter()
- Implementando una clase para iterar cadenas
- La función range()
- Generadores

Docstrings

- Documentando el código
- Ver la documentación en la consola
- Guardar la documentación en formato HTML
- Buscar documentación indicando una palabra clave

simultáneamente varias operaciones en el mismo espacio de proceso se...

Archivo

Archivo

python.org

pypi.org

Sitios

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

- Utilizar funciones en programas y acceder a su documentación

Anotaciones de tipos: typing

- Variables y constantes
- Funciones
- Clases
- Anotaciones para tipos complejos
- Alias
- Funciones con varios valores de retorno
- Anotaciones para variables con valores de distinto tipo
- Mypy

Convertir programas Python 2.x a 3.x

- Mostrar diferencias del código a convertir
- Traducir el código de Python 2.x a 3.x
- Aplicar o excluir reglas de traducción específicas
- Conversión de módulos de terceros

Variables locales y variables globales

Copia superficial y profunda de variables

- Identificadores de objetos
- Las funciones copy() y deepcopy() del módulo copy

Evaluar, ejecutar y compilar cadenas

- La función eval()
- La función exec()
- La función compile()

Programación Orientada a Objetos (I)

- Introducción
- Clases, Atributos y Métodos
- Características de la Programación Orientada a Objetos
- Variables de clases y variables de instancias
- Crear clases
- Crear objetos (instancias) de una clase
- Accediendo a los atributos y llamando a los métodos

Programación Orientada a Objetos (II)

- Funciones para atributos: getattr(), hasattr(), setattr() y delattr()
- Atributos de clase (Built-in)
- Destrucción de objetos (Recolección de basura)
- Herencia
- Herencia Múltiple
- Funciones issubclass() y isinstance()

Programación Orientada a Objetos (y III)

- Polimorfismo: Sobrecarga de Métodos
- Polimorfismo: Sobrecarga de Operadores
- Encapsulación: Ocultación de Datos
- Propiedades (properties)
- Orden de Resolución de Métodos (MRO).
- El atributo especial __mro__
- La función super()

Programación funcional. Funciones de orden superior

- La función map()
- La función filter()
- La función reduce()
- La función lambda
- Comprensión de listas
- Generadores
- La función Decorador

Enumeraciones

- Creación de una enumeración
- Comparaciones. La clase IntEnum
- Iteraciones
- Enumeraciones con constantes con valores únicos
- Enumeraciones con métodos

Data Classes: Clases de Datos

- Los métodos de dataclass
- Los parámetros de dataclass
- La función asdict()
- La función field()
- Herencia

Operaciones con archivos

- Abrir archivo
- Cerrar archivo
- Leer archivo: read, readline, readlines, with-as
- Escribir en archivo: write, writelines
- Mover el puntero: seek(), tell()
- Leer y escribir cualquier objeto a un archivo: pickle

Módulos

- Namespaces y Alias
- Creación y uso de un módulo
- La función dir()
- Paquetes (packages)

Diccionarios de variables locales y globales

- Funciones locals() y global()
- Llamadas de retorno. La función callable()

Acercamiento a la biblioteca estándar. Módulos os, sys y time

Explorando directorios con os.listdir, os.walk y os.scandir

- Obtener las entradas de un directorio: listdir()
- Obtener recursivamente las entradas de un directorio: walk()
- Obtener iterador con las entradas de un directorio: scandir()

Fileinput: procesar múltiples archivos fácilmente

- Procesar lista de archivos con fileinput.input()
- Procesar con funciones que ayudan a controlar el flujo
- Procesar lista de archivos obtenida con glob.glob()
- Procesar archivos incluidos como argumentos de un script
- Procesar múltiples líneas de la entrada estándar (stdin)
- Procesar lista de archivos con la sentencia with
- Procesar reescribiendo lista de archivos en la salida estándar
- La clase fileinput.FileInput()
- Procesar lista de archivos con la clase FileInput()
- Procesar lista de archivos con la clase FileInput() y readline()
- Procesar archivos con clase FileInput() con una codificación específica
- Procesar archivo comprimido con clase FileInput()

Filtrando archivos y directorios con glob y fnmatch

- El módulo Glob:
- Obtener entradas que coincidan con un patrón: glob.glob()
- Obtener iterador con entradas que coincidan con un patrón: glob.iglob()
- Adaptar cadenas con caracteres especiales: escape()
- El módulo fnmatch:
- Comprobar si una entrada coincide con un patrón: fnmatch.fnmatch()
- Comprobar si una entrada coincide diferenciando minúsculas y mayúculas: fnmatch.fnmatchcase()
- Obtener lista con entradas que coinciden con patrón: fnmatch.filter()
- Obtener patrón de búsqueda convertido en expresión regular: fnmatch.translate()

Copiar, mover y borrar archivos/directorios con shutil

- Copiar archivos completos o parciales: copyfileobj()
- Copiar archivos completos sin metadatos: copyfile()
- Copiar los permisos de un archivo: copymode()
- Copiar el estado de un archivo: copystat()
- Copiar archivos con permisos: copy()
- Copiar archivos con permisos y metadatos: copy2()
- Copiar un árbol de directorios con sus archivos: copytree()
- Borrar un árbol de directorios con sus archivos: rmtree()
- Mover un archivo o un directorio con su contenido: move()
- Obtener información del espacio de un disco: disk_usage()
- Cambiar propietario y/o grupo de un archivo/directorio: chown()
- Obtener la ruta de un archivo ejecutable: which()

Pathlib: rutas orientadas a objetos

- Rutas puras y concretas

Empaquetar y desempaquetar archivos con shutil

- Empaquetar un directorio: make_archive()
- Desempaquetar un directorio: unpack_archive()
- Obtener formatos permitidos para empaquetar: get_archive_formats()
- Obtener formatos permitidos para desempaquetar: get_unpack_formats()
- Otras funciones relacionadas

El módulo collections

- Observaciones

Archivos de configuración

- El módulo ConfigParser
- Crear secciones y opciones de configuración
- Guardar en un archivo la configuración
- Leer datos del archivo de configuración
- La sección DEFAULT

Solucionando errores con el depurador

- El módulo de depuración pdb
- Tipos de errores
- Uso de depurador

Facilitando la depuración de programas con breakpoint()

Rastreando la ejecución de un programa (trace)

- Rastrear la ejecución de un programa
- Rastrear las líneas que se ejecutan
- Rastrear, contar línea ejecutadas y obtener líneas no ejecutadas
- Rastrear llamadas a funciones
- Rastrear relaciones entre módulos por llamadas a funciones
- Rastrear y obtener informes combinados
- Agregar opción para rastrear en el editor de código

Probando el código con doctest

- Pruebas insertadas en el código fuente
- Prueba independientes del código fuente
- Escribiendo pruebas para doctest
- Opciones para modificar el comportamiento de doctest

Expresiones regulares. Módulo re

- La función match()
- Comodines
- Carácter especial
- Alternativas
- Grupos aislados
- Rangos
- Caracteres predefinidos
- Caracteres que permiten repeticiones
- Coincidencias al comienzo y al final
- El objeto mo y el método group()
- La función search()
- Métodos start() y end()
- La función findall()
- La función finditer()
- La función split()
- La función sub()

Difflib: encontrando las diferencias

- Clase Differ. Comparando secuencias
- Obtener diferencias entre textos con Differ() y compare()
- Obtener diferencias entre textos con ndiff()
- Ajustar los resultados de ndiff() con linejunk y charjunk
- Obtener diferencias con restore()
- Obtener diferencias unificadas con unified_diff()
- Obtener las mejores coincidencias con get_close_matches()
- Obtener diferencias entre códigos HTML con HtmlDiff()
- La clase difflib.SequenceMatcher
- Obtener nivel de coincidencias (ratio) entre dos cadenas
- Obtener tuplas que describen pasos para convertir *a* en *b*
- Obtener las posiciones con las coincidencias de dos secuencias

Filecmp: comparando archivos y directorios

- Comparar dos archivos: cmp()
- Comparar lista de archivos en dos directorios: cmpfiles()
- La clase dircmp

Empaquetado y distribución de proyectos Python (I)

- PyPI, pip y PyPA
- Paquete de distribución: paquetes source o sdist y paquetes wheel
- Requerimientos para instalar paquetes de distribución
- Uso de pip

Empaquetado y distribución de proyectos Python (y II)

- Introducción

- Requerimientos
- Caso práctico de empaquetado y distribución de un proyecto
- Empaquetar el proyecto
- Subir el proyecto a PyPI
- Herramientas
- El modo 'desarrollo'

Operaciones con archivos CSV

- Leer archivo CSV con reader()
- Leer archivo CSV con reader() y realizar algunas operaciones básicas
- Ordenar datos con sort(), sorted() e itemgetter().
- Leer archivo CSV con reader() y ordenar salida por un campo
- Leer archivo CSV con DictReader() y filtrar salida
- Leer archivo CSV con DictReader() y consultar propiedades
- Leer archivo CSV con DictReader() y ordenar salida por nombre de campo
- Escribir archivo CSV con writer() y writerow(). Quoting
- Escribir archivo CSV con writer() y writerows()
- Leer archivo CSV con reader(), skipinitialspace y strict
- Escribir archivo CSV con DictWriter() capturando excepciones
- Leer archivo CSV con reader() de un dialecto determinado
- Crear dialecto y leer archivo CSV
- Listar dialectos disponibles
- Obtener información de un dialecto
- Suprmir dialecto
- Deducir con Sniffer() el dialecto de un archivo csv
- Deducir con Sniffer() si un archivo tiene encabezado de campos
- Mostrar/Establecer límite de tamaño de campo a analizar

Buscar, extraer y depurar datos con Pysaurio

- Plantillas .rap
- Secciones de una plantilla .rap
- Funciones para reglas (rules)
- Métodos de la clase Pyraptor
- Generar archivo .CSV (campos en filas)
- Generar archivo .CSV (campos en columnas)
- Generar varios archivos .CSV utilizando varias plantillas .rap

Base de datos SQLite3

- Crear base de datos
- Crear tabla
- Iniciar y cerrar una conexión con la base de datos
- Declarar y cerrar un cursor
- Añadir registro a la tabla
- Completar la transacción de inserción de registro
- Consultar todos los registros de la tabla
- Consultar registro con parámetro
- Consultar los registros, uno a uno, con fetchone()
- Consultar un nº de registros concretos con fetchmany()
- Consultar todos los registros, con fetchall()
- Mostrar los campos de un registro por su posición
- Consultar registros por nombre de campos
- Borrar registro
- Modificar campos
- Deshacer modificación

Base de datos SQLite con APSW

- Instalar el módulo APSW
- Comprobar versión instalada
- Crear y/o Abrir una base de datos SQLite
- Ejecutar comandos SQL
- Crear tablas
- Listar campos de una tabla
- Insertar registros
- Consultar registros
- Encadenar varias consultas
- Obtener la descripción de los campos
- Habilitar restricciones de integridad referencial
- Borrar registros
- Crear una vista
- Insertar registros utilizando bindings
- Trazar la ejecución de comandos
- Ejecutar varias sentencias

Bases de datos MySQL (y MariaDB) con PyMySQL

- Instalar PyMySQL con PIP
- Conectar con base de datos y obtener datos
- Añadir una tabla nueva
- Insertar registros en la tabla nueva
- Mostrar tablas de la base de datos
- Borrar una tabla

Bases de datos documentales con TinyDB (I). Operaciones básicas

- Introducción
- Instalación de TinyDB
- Importar módulos para realizar operaciones básicas
- Abrir/Crear una base de datos. La tabla por defecto ("_default")
- Abrir/Crear una tabla con un nombre determinado
- Obtener los nombres de las tablas de una base de datos
- Insertar registros en una tabla
- Consultar todos los registros de una tabla
- Consultar todos los registros que cumplan una condición
- Consultar el primer registro que cumpla una condición
- Actualizar el dato de un registro que cumpla una condición
- Suprimir los registros que cumplan una condición
- Mostrar contenido de la base de datos desde la consola
- Borrar todos los registros de una tabla
- Borrar todas las tablas de la base de datos
- Cerrar una base de datos

Bases de datos documentales con TinyDB (y II). Uso avanzado

- Insertar múltiples registros
- Insertar registros con campos diferentes
- Insertar un nuevo campo en un registro en una actualización
- Insertar múltiples registros insertando campo con valor autoincrementado
- Actualizar borrando un campo de un registro que cumpla una condición
- Actualizar incrementando en 1 el valor numérico de un campo
- Actualizar decrementando en 1 el valor numérico de un campo
- Obtener el número de registros que tiene una tabla
- Leer un registro que cumpla una condición (get)
- Conocer si hay registros que cumplan una condición
- Contar los registros que cumplan una condición
- Consultar un registro por su ID
- Obtener el ID de un registro que cumpla una condición
- Actualizar un campo de varios registros por sus ID
- Borrar varios registros por su ID
- Consultar registros que no cumplan una condición (NOT -> "~")
- Consultar registros que cumplan una condición u otras (OR -> "|")
- Consultar registros que cumplan más de una condición (AND -> "&")
- Consultar registros utilizando expresiones regulares
- Consultar registros aplicando funciones
- Insertar registros con campos anidados (diccionarios)
- Consultar registros con campos anidados (diccionarios)
- Insertar registros con campos anidados (listas)
- Consultar registros con campos anidados (listas) que al menos uno cumpla
- Consultar registros con campos anidados (listas) que todos cumplan
- Consultar registros con campos anidados (listas) que contengan valor
- Crear base de datos utilizando la memoria convencional

Ejecutar programa con argumentos

Ejecutar un comando externo

- Ejecutar comando con os.system()
- Ejecutar comando con os.open() y capturar su salida
- Ejecutar comando con subprocess.call(), capturar salida y resultado
- Ejecutar comando con subprocess.Popen()
- Ejecutar comando con subprocess.Popen(), capturar salida y errores
- Ejecutar comando con subprocess.Popen(), capturar salida en tiempo real

Cmd: Construyendo un intérprete de comandos

- La clase Cmd. Métodos de la clase Cmd
- Entrando en bucle: cmdloop()
- Ejecutar comandos, uno a uno: onecmd()
- Línea sin comando: emptyline()
- Comando erróneo: default()
- Antes y después de ejecutar un comando: precmd() y postcmd()

- Al iniciar y terminar el bucle: preloop() y postloop()
- Atributos de la clase Cmd
- Analizar comandos que incluyen argumentos

Advertencias con el módulo warnings

Generar un archivo log

Capturar usuario actual e introducir contraseña (no visible)

El módulo time

- Funciones para obtener y convertir tiempo (civil)
- Obtener tiempo local (fecha-hora) en segundos: time()
- Convertir a cadena tiempo en segundos: ctime()
- Obtener tiempo UTC a partir de segundos: gmtime()
- Obtener tiempo local como objeto struct_time: localtime()
- Convertir a cadena un objeto struct_time: asctime()
- Convertir a segundos un objeto struct_time: mktime()
- Funciones para validar y aplicar formatos
- Validar tiempo expresado como cadena: strptime()
- Convertir y formatear un objeto struct_time: strftime()
- Funciones y variables para obtener información del huso horario
- altzone, timezone(), tzname, time.tzset() y time.daylight
- Funciones para obtener tiempo del sistema
- monotonic(), perf_counter(), process_time() y get_clock_info()
- Función para detener un programa: sleep()

Operaciones con fechas y horas (datetime). Calendarios

- Mostrar fecha y hora (datetime)
- Comparando fechas y horas (datetime, date)
- Aplicando formatos a fechas y horas (Máscaras)
- Para convertir una cadena a objeto datetime
- Operaciones con fechas y horas
- Otros ejemplos de operaciones con otras unidades de tiempo
- Diferencia entre dos fechas (datetime)
- Diferencia entre dos fechas en días (datetime y strptime)
- Diferencia de dos fechas (date)
- Expresar una fecha en formato largo
- A partir de una fecha se obtiene tupla con año, nº semana y día de semana
- Obtener día de la semana por su número
- Dado el ordinal se obtiene la fecha correspondiente
- Dada una fecha se obtiene un ordinal (01-01-0001 -> 1)
- Obtener una tupla a partir de fecha-hora (datetime)
- Convertir un ordinal en fecha-hora (fromtimestamp)
- Obtener calendario del mes actual (calendar.month)
- Obtener calendario del mes actual (calendar.TextCalendar)
- Obtener matriz con calendario de mes actual: (Calendar monthdayscalendar)
- Obtener matriz de tuplas con calendario: (Calendar monthdays2calendar)
- Calendario completo del año

Programar eventos para ejecutar tareas

- El módulo sched
- Declarar un programador: scheduler()
- Programar eventos y poner en marcha el programador: run()
- Programar eventos para ejecutar en un momento determinado: entertabs()
- Programar eventos con posibilidad de cancelación: cancel()
- Programar eventos para ejecutar después de un tiempo de espera: enter()
- Programar eventos para ejecutar considerando la prioridad
- Programar eventos para ejecutar controlando el bloqueo: blocking

Threading: programación con hilos (I)

- Objetos Thread: los hilos
- Hilos con argumentos
- Hilos que funcionan durante un tiempo
- Demonios
- Controlar la ejecución de varios demonios
- Crear una subclase Thread y redefinir sus métodos

Threading: programación con hilos (y II)

- Temporizadores (Timer).
- Sincronizar hilos con objetos Event
- Control de acceso a los recursos. Bloqueos
- Sincronizar hilos con objetos Condition
- Sincronizar hilos con objetos Barrier

- Imprimir tablas con distintos formatos
- Alineación y formato
- Tabulate desde la línea de comandos

Ptpython: efectivo, cómodo y funcional

El depurador PuDB

Proyectos de documentación con MkDocs

- Instalación de MkDocs
- Creando un proyecto nuevo de documentación
- Examinando los documentos iniciales
- Comenzando a trabajar con el servidor de desarrollo
- La barra de navegación
- Cambiando la apariencia del proyecto
- Construyendo el sitio

Pillow

Fundamentos para procesar imágenes con Pillow (I)

- Instalación
- Abrir imágenes
- Consultar información de las imágenes
- Edición básica (tamaños, rotar, girar...) y guardar imágenes

Fundamentos para procesar imágenes con Pillow (II)

- Tipos de imágenes y formatos
- Convertir imágenes
- Filtros

Fundamentos para procesar imágenes con Pillow (y III)

- Añadir bordes a una imagen o recortarlos
- Aplicar efectos
- Dibujar sobre una imagen
- Escribir texto sobre una imagen

EasyGUI

- EasyGUI, la interfaz gráfica fácil
- Mostrar cajas de mensajes
- Entrada de datos
- Entrada de contraseñas
- Listas de opciones
- Gestión de errores
- Ventanas comunes: abrir directorio, abrir archivo y guardar archivo
- Egstore: archivos de configuración

Tkinter

Interfaces gráficas en Python

- Introducción
- Consultar la versión de Tkinter
- Instalar Tkinter
- La primera aplicación
- La primera aplicación, orientada a objetos
- Obtener información de una ventana

Diseñando ventanas gráficas

- Introducción
- El gestor de geometría Pack
- El gestor de geometría Grid
- El gestor de geometría Place

Tipos de ventanas

- Ventanas de aplicación y de diálogos

- Ventanas modales y no modales

Variables de control

- Declarar variables de control
- Método set()
- Método get()
- Método trace()
- Estrategias para validar y calcular datos

Menús, barras de herramientas y de estado

- Introducción
- Menús de opciones
- Barras de herramientas
- Barra de estado
- PyRemoto, un ejemplo de implementación

Inicio

Suscribirse a: Entradas (Atom)