Python 3 para impacientes

"Simple es mejor que complejo" (Tim Peters)

miércoles, 26 de abril de 2017

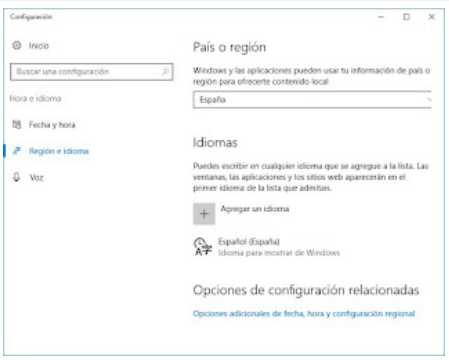
Internacionalización del código (I)

El módulo locale

Cuando se instala un sistema operativo una de las primeras opciones de configuración que se decide es la **localización geográfica** del equipo o dispositivo. Normalmente, el idioma del sistema operativo se corresponderá con el país seleccionado y dicha elección será también empleada para fijar una serie de convenciones que definen para ese lugar el modo de expresar los números, las cantidades monetarias, los números positivos y negativos, las fechas y horas, etc.

Estas convenciones establecen, entre otros, el carácter que se utiliza en los números para separar la parte entera de la decimal (",", ".", etc.); el carácter que se usa como separador de millares (".", ",", etc.), el símbolo monetario (€, £, $, ¥, etc.), el formato de las fechas y horas, etc.

Afortunadamente, estas definiciones pueden ser utilizadas (y cambiadas) por una aplicación para adaptar la forma en que presentan sus datos a cada territorio. Particularmente, en Python ese es el cometido principal que tiene el módulo **locale**, incluido en la librería estándar como parte del soporte que ofrece el lenguaje a la internacionalización de las aplicaciones.



Herramienta de Configuración Regional y de Idioma

Hoy, los distintos sistemas operativos proporcionan herramientas gráficas que permiten modificar de forma fácil las opciones comentadas. En Windows se utiliza la herramienta de "**Configuración Regional y de Idioma**" y en GNU/Linux ésta, a veces, se denomina de igual forma (como en Ubuntu GNOME) o puede estar integrada en otra herramienta como la de "**Soporte de Idiomas**" (como sucede en Xubuntu).

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...

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...

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...

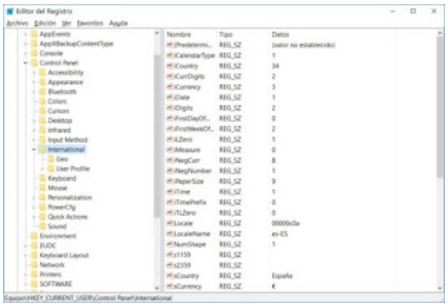
Tkinter: interfaces gráficas en Python
Introducción Con Python hay muchas posibilidades para programar una interfaz gráfica de usuario (GUI) pero Tkinter es fácil d...

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



Configuración Regional en Registro de Windows

Según sea el caso, en Windows las definiciones vigentes del entorno de un usuario son almacenadas en el **Registro de Windows** (en la rama "HKEY_CURRENT_USER*Control Panel**International*") y en un sistema GNU/Linux en los archivos donde se declaran las variables de entorno (**LC_ALL**, **LC_CTYPE**, **LANG**, **LANGUAGE**) que son ejecutados al iniciar un equipo. Cada plataforma tiene sus propios mecanismos.

En GNU/Linux para consultar desde la línea de comandos las variables de entorno utilizar los comandos **env** o **printenv**. En Windows, el comando **systeminfo** muestra información básica de la configuración regional.

Obtener/establecer la configuración regional: setlocate()

La función **setlocate()** devuelve o establece la configuración regional del entorno de un usuario. El argumento **category** representa una categoría determinada de configuración (una categoría agrupa un conjunto de definiciones de configuración). Y **locale** es una cadena que representa el idioma de una localización geográfica determinada:

locale.setlocale(category, locale=None)

Las definiciones de configuración se agrupan en las siguientes categorías:

- locale.LC_ALL: combinación de todas las categorias.
- locale.LC_NUMERIC: formatos de números.
- locale.LC_TIME: formatos de fechas/horas.
- locale.LC_MONETARY: formatos de valores monetarios.
- locale.LC_COLLATE: para ordenación de cadenas.
- locale.LC_CTYPE: para funciones de tipo de carácter.
- locale.LC_MESSAGES: para visualización de mensajes. (Python no admite mensajes específicos)

Ver valores del argumento **locale** en Windows

En GNU/Linux Ubuntu se pueden consultar los idiomas disponibles con el comando **locale -a**.

También, se pueden instalar más idiomas con el comando **sudo apt-get install language-pack-XXX**. Por ejemplo, para instalar la especificación de Japón: **sudo apt-get install language-pack-ja**. (Es equivalente a utilizar la herramienta de "**Soporte de Idiomas**").

A continuación, varios ejemplos para fijar la configuración regional en todas las categorías (**locale.LC_ALL**). Si la cadena del idioma es incorrecta o el idioma no está disponible se producirá la siguiente excepción:

Error: unsupported locale setting.

Cuando se desarrolla una aplicación para que funcione en cualquier país teniendo en cuenta la configuración regional, después de importar el módulo **locale** se establecerá la configuración que tenga el entorno de trabajo del propio usuario. Para ello, el argumento **locale** de **setlocate()** debe ser una cadena vacía: ''

Así, si un usuario en España imprime importes utilizando la configuración regional, el símbolo monetario mostrado será el símbolo del euro '€'; si es un londinense, el símbolo de la libra '£'; un japonés, el símbolo del Yen '¥', etc.

```
import locale

# Establecer la configuración que tenga el entorno del usuario
locale.setlocale(locale.LC_ALL, '')

# Establecer configuraciones de países concretos:

# Establecer configuración para España en un sistema Windows
locale.setlocale(locale.LC_ALL, 'esp')

# Establecer configuración para España en Ubuntu
```

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

Archivo

abril 2017 (1)

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

```
locale.setlocale(locale.LC_ALL, 'es_ES.utf8')

# Establecer configuración para España en otros sistemas GNU/Linux
locale.setlocale(locale.LC_ALL, 'es_ES')

# Establecer configuración para Japón en Ubuntu
locale.setlocale(locale.LC_ALL, 'ja_JP.utf8')

# Establecer configuración para Japón en otros sistemas GNU/Linux
locale.setlocale(locale.LC_ALL, 'ja_JP')
```

Obtener diccionario con configuración actual: locale.localeconv()

La función **localeconv()** devuelve un diccionario con todas las definiciones de la configuración regional actual.

```
import locale
import pprint

# Establecer la configuración del entorno de usuario
# En este ejemplo se corresponde con 'es_ES.utf8' (España)
locale.setlocale(locale.LC_ALL, '')

# Obtener definiciones de la configuración actual
configuracion = locale.localeconv()

# Imprimir definiciones con pprint para una
# lectura agradable:
imprimir = pprint.PrettyPrinter()
imprimir.pprint(configuracion)

'''
Salida para España:

{'currency_symbol': '€',
 'decimal_point': ',',
 'frac_digits': 2,
 'grouping': [3, 3, 0],
 'int_curr_symbol': 'EUR ',
 'int_frac_digits': 2,
 'mon_decimal_point': ',',
 'mon_grouping': [3, 3, 0],
 'mon_thousands_sep': '.',
 'n_cs_precedes': 0,
 'n_sep_by_space': 1,
 'n_sign_posn': 1,
 'negative_sign': '-',
 'p_cs_precedes': 0,
 'p_sep_by_space': 1,
 'p_sign_posn': 1,
 'positive_sign': '',
 'thousands_sep': '.'}
'''

# Establecer configuración para Japón
# en Ubuntu.
locale.setlocale(locale.LC_ALL, 'ja_JP.utf8')
# (En Windows utilizar la cadena 'jpn')

# Obtener definiciones de Japón
configuracion = locale.localeconv()

# Imprimir definiciones con pprint para una
# lectura agradable
imprimir.pprint(configuracion)

'''
Salida para Japón:

{'currency_symbol': '￥',
 'decimal_point': '.',
 'frac_digits': 0,
 'grouping': [3, 0],
 'int_curr_symbol': 'JPY ',
 'int_frac_digits': 0,
 'mon_decimal_point': '.',
 'mon_grouping': [3, 0],
 'mon_thousands_sep': ',',
 'n_cs_precedes': 1,
```

```
 'n_sep_by_space': 0,
 'n_sign_posn': 4,
 'negative_sign': '-',
 'p_cs_precedes': 1,
 'p_sep_by_space': 0,
 'p_sign_posn': 4,
 'positive_sign': '',
 'thousands_sep': ','}
'''
```

Funciones para aplicar las definiciones de configuración

Para aplicar las definiciones de una configuración regional utilizar las siguientes funciones:

```
import locale
locale.setlocale(locale.LC_ALL, '')

# Formatear números de acuerdo con la configuración
# actual de LC_NUMERIC
valor = 123456.78
locale.format('%10.2f', valor, grouping=True)   # '123.456,78'
locale.format('%10.2f', valor, grouping=False)  # ' 123456,78'

# Formatear procesando especificadores de formato %
locale.format_string('%i', valor, grouping=True)  # '123.456'
locale.format_string('%.1f', valor, grouping=True)  # '123.456,8'

# Formatear números de acuerdo con la
# configuración LC_MONETARY actual.
locale.currency(valor, symbol=True, grouping=False)  # '123456,78 €'
# Establecer configuración para Reino Unido
locale.setlocale(locale.LC_ALL, 'en_IN')
locale.currency(valor, symbol=True, grouping=False)  # '₹ 123456.78'
# Establecer configuración predeterminada
locale.setlocale(locale.LC_ALL, '')

# Formatear flotantes como str pero con el carácter
# predeterminado para el punto decimal
locale.str(valor)  # '123456,78'

# Comparar dos cadenas de acuerdo con la
# configuración LC_COLLATE actual
locale.strcoll('Guadalquivir', 'Guadalhorce')  # 10
locale.strcoll('Guadalquivir', 'Guadalupe')  # -4
locale.strcoll('Guadalupe', 'Guadalupe')  # 0

# Definir clave de ordenación de acuerdo a la
# configuración actual
cadenas = ['Guadalquivir', 'Guadalhorce', 'Guadalupe']
cadenas.sort(key=locale.strxfrm)
print(cadenas)  # ['Guadalhorce', 'Guadalquivir', 'Guadalupe']

# Convertir cadena en cadena numérica normaliza
# de acuerdo con LC_NUMERIC.
# Disponible a partir de Python 3.5
cadena = '6543,21'
locale.delocalize(cadena)  # '6543.21'
```

Obtener información específica de configuración: locale.nl_langinfo()

La función **nl_langinfo()** retorna una cadena con información específica de la configuración regional actual. Esta función no está disponible en todos los sistemas y el conjunto de opciones posibles también puede variar entre plataformas.

```
import locale
from datetime import datetime

# Establecer configuración del entorno de usuario
# En este ejemplo se corresponde con 'es_ES.utf8' (España)
locale.setlocale(locale.LC_ALL, '')

# Obtener codificación de los caracteres utilizada
codificacion = locale.nl_langinfo(locale.CODESET)  # 'UTF-8'

# En este caso la codificación se utiliza para
# convertir una cadena de tipo Unicode a Byte
```

```
pais = bytes("España", codificacion)
print(pais)  # b'Espa\xc3\xb1a'
# El formato Byte acepta sólo caracteres ASCII y
# facilita la creación de archivos de texto.

# Obtener formato de fecha y hora
formato1 = locale.nl_langinfo(locale.D_T_FMT)  # '%a %d %b %Y %T %Z'

# Obtener fecha y hora actual e imprimir utilizando
# el formato de fecha-hora de la configuración actual
hoy = datetime.today()
print(hoy.strftime(formato1)) # mié 26 abr 2017 17:24:42

# Obtener formato de fecha e imprimir utilizando
# el formato de fecha de la configuración actual
formato2 = locale.nl_langinfo(locale.D_FMT)  # '%d/%m/%y'
print(hoy.strftime(formato2)) # 26/04/17

# Obtener formato de hora e imprimir utilizando
# el formato de hora de la configuración actual
formato3 = locale.nl_langinfo(locale.T_FMT)  # '%T'
print(hoy.strftime(formato3))  # 17:24:42

# Obtener cadena para representar la hora con
# el formato AM/PM.
# Si devuelve cadena vacía se expresa con 24 horas
formato4 = locale.nl_langinfo(locale.T_FMT_AMPM)  # ''

# Obtener nombre completo y abreviado de los días de
# la semana en el idioma de la configuración actual
for dia in range(1, 8):
    nombre_dia = 'locale.DAY_' + str(dia)
    abrev_dia = 'locale.ABDAY_' + str(dia)
    print(nombre_dia, ':',
          locale.nl_langinfo(eval(nombre_dia)),
          '-',
          locale.nl_langinfo(eval(abrev_dia)))

'''
Salida:

locale.DAY_1 : domingo - dom
locale.DAY_2 : lunes - lun
locale.DAY_3 : martes - mar
locale.DAY_4 : miércoles - mié
locale.DAY_5 : jueves - jue
locale.DAY_6 : viernes - vie
locale.DAY_7 : sábado - sáb
'''

# Obtener nombre completo y abreviado de los meses
# en el idioma de la configuración actual
for mes in range(1, 13):
    nombre_mes = 'locale.MON_' + str(mes)
    abrev_mes = 'locale.ABMON_' + str(mes)
    print(nombre_mes, ':',
          locale.nl_langinfo(eval(nombre_mes)),
          '-',
          locale.nl_langinfo(eval(abrev_mes)))

'''
Salida:

locale.MON_1 : enero - ene
locale.MON_2 : febrero - feb
locale.MON_3 : marzo - mar
locale.MON_4 : abril - abr
locale.MON_5 : mayo - may
locale.MON_6 : junio - jun
locale.MON_7 : julio - jul
locale.MON_8 : agosto - ago
locale.MON_9 : septiembre - sep
locale.MON_10 : octubre - oct
locale.MON_11 : noviembre - nov
locale.MON_12 : diciembre - dic
'''

# Obtener carácter que se emplea en los números
# para separar la parte entera de la decimal de
# la configuración actual
locale.nl_langinfo(locale.RADIXCHAR)  # ','
```

```
# Obtener carácter separador de millares
locale.nl_langinfo(locale.THOUSEP)  # '.'

# Obtener expresión regular que se puede usar con la función regex
# para reconocer una respuesta positiva o negativa a una pregunta.
locale.nl_langinfo(locale.YESEXPR)  # '^[sSyY].*'
locale.nl_langinfo(locale.NOEXPR)  # '^[nN].*'

# Obtener símbolo monetario
# El signo '-' indica que el símbolo aparecerá delante del valor
# El signo '+' indica que el símbolo aparecerá detrás del valor
# El '.' indica que el símbolo aparecerá en la posición del
# punto o coma decimal
locale.nl_langinfo(locale.CRNCYSTR)  # '+€'
```

Funciones para obtener el idioma y la codificación

Las funciones siguientes permiten obtener el idioma y la codificación de las cadenas de texto de distintos modos:

```
import locale
locale.setlocale(locale.LC_ALL, '')

# Determinar configuración a partir de variable de entorno
locale.getdefaultlocale()  # ('es_ES', 'UTF-8')
locale.getdefaultlocale(['LANG'])  # ('es_ES', 'UTF-8')

# Obtener configuración del entorno local a partir de categoria
locale.getlocale()  #  ('es_ES', 'UTF-8')
locale.getlocale(locale.LC_NUMERIC)  # ('es_ES', 'UTF-8')

# Obtener codificación que se utiliza para representar
# las cadenas de texto
locale.getpreferredencoding()  # 'UTF-8'

# Obtener código de configuración regional normalizado
locale.normalize('es_ES')  #  'es_ES.ISO8859-1'

# Establecer configuración regional para la categoria indicada
locale.resetlocale(locale.LC_NUMERIC)
```

Ir al índice del tutorial de Python

Publicado por Pherkad en 16:18

Etiquetas: Python3

Entrada más reciente Inicio Entrada antigua