

# Demo ip\_lib

## Procesa informes de salida CSV para una aplicación de gestión

**ip\_lib** es un módulo Python, cuyo propósito es facilitar la integración a un ERP o aplicación de gestión, de datos estructurados en informes CSV de cierta complejidad, es **de uso específico** para estos informes. Existe experiencia exitosa del desarrollo de módulos para Oddo/OpenERP actualmente en producción.

ip\_lib es [open source](#).

### Requerimientos python

```
logging
simplejson
sqlite3
ipdb          (opcional)
```

### Instalación:

En una terminal ejecutar:

```
$ wget https://github.com/infoprismo/demo_spazos/archive/refs/heads/main.zip
$ unzip main.zip
$ cd demo_spazos
$ ipython -i demo.py
```

### Lo que viene

En el prompt del intérprete python >>>, quedará disponible un diccionario *informes* con dos llaves de acceso a colecciones de instancias de objetos en ésta demo.

```
>>> informes.keys()
['jornadas', 'lote']
```

### La llave 'jornadas'

A través de la llave 'jornadas' se accede a un diccionario. Las llaves de acceso en éste se corresponden a las jornadas de operaciones de los informes leídos.

Vemos los elementos de jornadas:

```
>>> informes['jornadas'].keys()
[2021-07-10,
 2021-07-08,
 2021-07-09,
 2021-07-04,
 2021-07-05,
 2021-07-06,
 2021-07-07,
 2021-07-01,
 2021-07-02,
 2021-07-03]
```

La colección de tickets de una jornada específica:

```
>>> informes['jornadas']['2021-07-01']
[Ticket1,
 Ticket2,
 ...
 TicketN]
```

Donde "Ticket1", Ticket2 etc, son los tickets del informe correspondiente al día 2021-07-01. Esta es la sintaxis para acceder al primer elemento de la lista, que es instancia de una clase definida en *ip\_lib*, en este ejemplo corresponde al primer ticket de la jornada.

```
>>> informes['jornadas']['2021-07-01'][0]
```

Asignando el objeto/instancia a una variable:

```
>>> tck1 = informes['jornadas']['2021-07-01'][0]
```

a través de *tck1* accedemos a datos y métodos disponibles.

Los atributos de tck1 *cabecal* y *lineas* son a su vez instancias que modelan respectivamente a las líneas de cabecal y a las líneas de detalle, según definiciones contenidas en el documento **Diseño de la salida estándar** de la aplicación que genera los informes CSV, Versión: 1.6.79 Los datos que albergan las instancias de éstos objetos siguen el nomenclátor -lowrecase, de dicha especificación.

Acceso al número de ticket:

```
>>> tck1.cabecal.numero_tck
292197
```

La instancia accede directamente a cada uno de los atributos de cabecal o a todos en un único diccionario.

```
>>> tck1.cabecal.zip
cantidadarticulos: '25',
cantidadlineas: '30',
codigocaja: '001',
codigocajera: '10',
date: '20210701',
descripcion: 'Ticket Venta/Ingreso',
estadoticket: 'F',
fecha: '2021-07-01',
numeroticket: '292197',
sucursal: '1',
timestamp_tck: '2021-07-01 11:46:23',
timestampticket: '20210701114623',
tipocabecal: '1',
tipocliente: '20',
totalapagar: '1593.04'
```

Por su parte *lineas* es un diccionario de objetos. Cada una de sus llaves de acceso se corresponde a su número de línea de detalle en el ticket:

```
>>> tck1 = informes['jornadas']['2021-07-01'][0]
>>> lin1 = tck1.lineas[1]
```

```
>>> lin1.tipolinea
'5'
```

```
>>> lin1.descripcion
'Cabecal de CFE'
```

```
>>> lin1.datos *
ciudadreceptor': ''
descripcioncfe': 'e-Ticket'
direccionreceptor': ''
documentoreceptor': ''
name': '2021070111462300100292197-5-1'
nombrereceptor': ''
numerocfe': '0685310'
seriecfe': 'B'
tipocfe': '101'
tipodocumentoreceptor': '0'
```

\* lin1.datos es análogo a tck.cabecal.zip

Un método especial disponible en *lineas* de cabecal o de detalle:

```
>>> tck1.cabecal.rlinea
C#1#1#292197#10#20210701114623#F#25#1593.04#20#30
```

```
>>> lin1.rlinea
L#1#5#114623#101#e-Ticket#B#0685310#0####
```

La llave '*lote*' ofrece un diccionario accesible por fecha, 'yyyyDDmm', igual al de la llave 'jornadas':

```
>>> informes['lote']
[2021-07-10,
 2021-07-08,
 2021-07-09,
 2021-07-04,
 2021-07-05,
 2021-07-06,
 2021-07-07,
 2021-07-01,
 2021-07-02,
 2021-07-03]
```

```
>>> informes['lote']['2021-07-10'].keys()
['tickets_pazos', 'sufigo', 'sucursal']
```

```
>>> informes['lote']['2021-07-10']['sufigo']
'2013'
```

```
>>> informes['lote']['2021-07-01']['sucursal']
'1'
```

La demo contiene una función predefinida:

```
»»» repazos_csv(jornadas, dia)
```

donde *dia* es una string de la forma 'yyyy-MM-dd' y '*jornadas*' la instancia del conjunto de informes leídos.

Haciendo uso de los métodos *rlinea*, la función reconstruye el informe original completo correspondiente a esa fecha o cualquier parte del mismo, sea un cabecal, una línea, como se vió más arriba, uno o varios tickets en particular, etc.

Extracto de 'repazos\_csv':

```
def repazos_csv(jornadas, dia=None):
    '''
    :param: jornada: instancia: datos y métodos de una jornada de operaciones 'jornadas='
    :param: dia:      text:      fecha de la jornada de la forma 'yyyyMMdd'
    :result: text:     bool:     True si se reconstruyó el infome, sino False
    '''

    tickts_jornada = informes['jornadas'][dia]

    info_csv = ''
    for t in tickts_jornada:
        info_csv += ''.join(t.cabecal.rlinea + '\n')
        for l in t.lineas:
            info_csv += ''.join(t.lineas[l].rlinea + '\n')
```

info\_csv contiene una string con un informe compelo Salidapazosnuevo\*

En esta demo la función reconstruye el informe completo, pudiendo optar por mostrarlo en pantalla o escribir su contenido en un archivo. Esta facilidad es básicamente de uso en debug.

*ip\_lib* puede escribir un archivo json con los nombres y valores de los datos informe. Implementa además el mantenimiento de una base de datos sqllite de un histórico de informes leídos y serializados en archivos json.

Nota: hay algunos de informes para pruebas en *ot/inout/informes*

Ideas, dudas, [errores](#).

demo en [github](#).