Documentación de Programas

Una buena **documentación de software**, ya sea un d**ocumento de especificaciones** para programadores y ensayadores, un **documento técnico** para usuarios internos, o **manuales de software** y **documentos de ayuda** para usuarios finales, ayuda a que la persona que trabaja con el software entienda sus características y funciones.

Una buena documentación de usuario es específica, concisa, y relevante, y le brinda toda la información importante a la persona que utiliza o modifica el software.

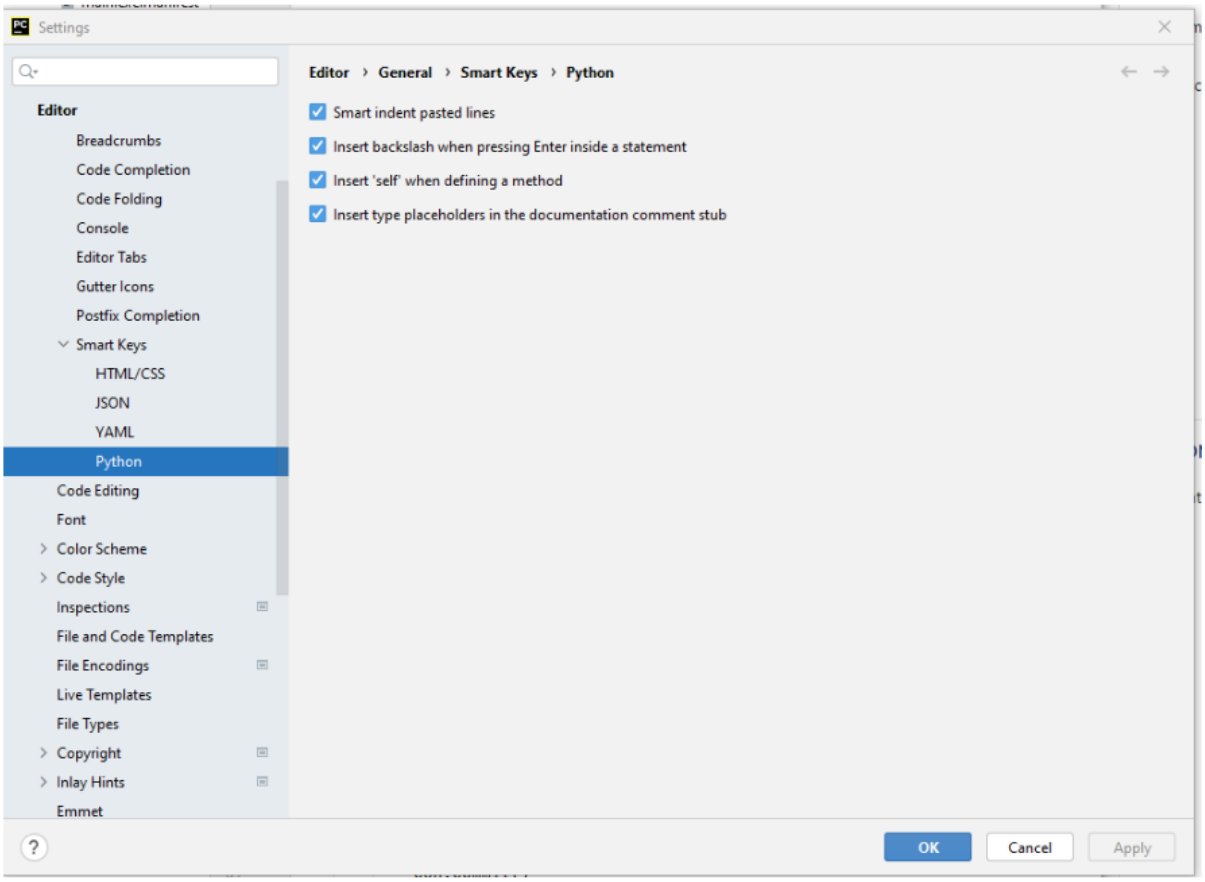
La **documentación de los programas** es un aspecto sumamente importante, ya no solo en el **desarrollo de la aplicación** sino **en el mantenimiento** de esta, en especial, en las probables modificaciones posteriores para su mejora que a veces no son realizadas por el mismo equipo de desarrollo.

La documentación se divide en dos categorías, interna y externa. **Interna** es la que se crea en el mismo código, ya puede ser en forma de comentarios o de archivos de información dentro de la aplicación y **externa,** la contenida en documentos o libros o como en la actualidad en soporte electrónico.

En primer lugar vamos a trabajar en la **documentación interna** del programa para luego, estructurar la documentación externa, concretamente en la elaboración de un **sencillo manual de usuario**.

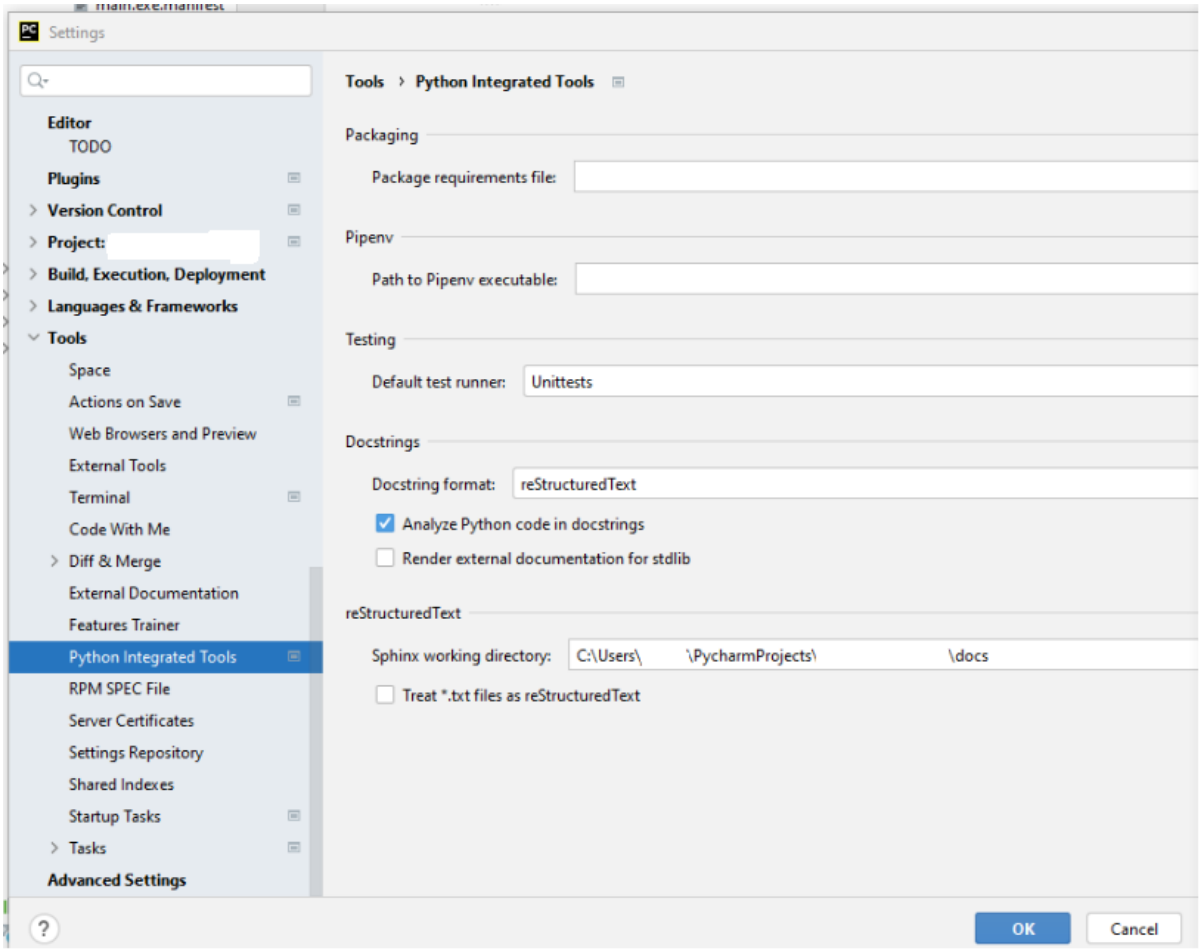
Documentación de programas en Python.

En primer lugar, preparamos PyCharm. En *File -> Settings -> Editor -> General -> Smart Keys -> Python* y clickeamos en *Insert type placeholders in the documentation comment stub*.



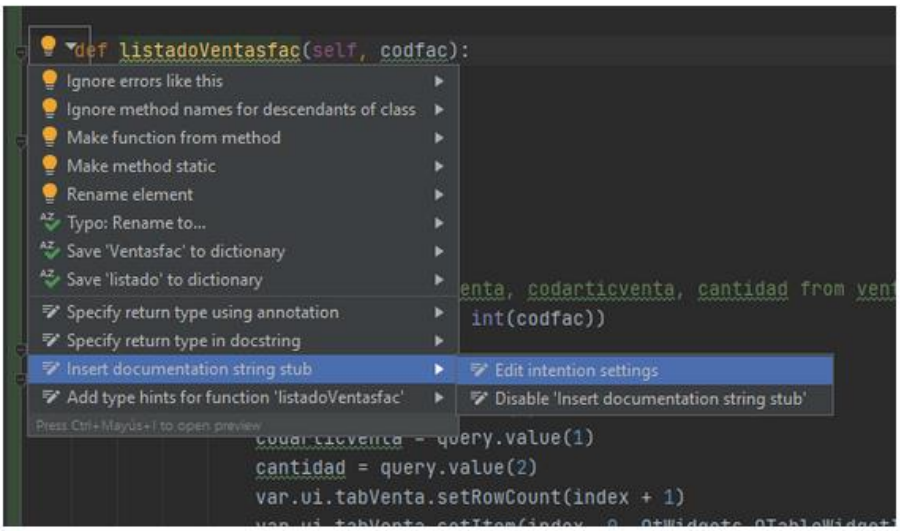
A continuación elegimos un generador de documentación. Hay varios con sus ventajas e inconvenientes, **docstrings** (el primero y más básico), **epydoc** o **reStructuredText.** Elegimos **reStructuredText,** que es un lenguaje de marcas y el más utilizado en la comunidad Python.

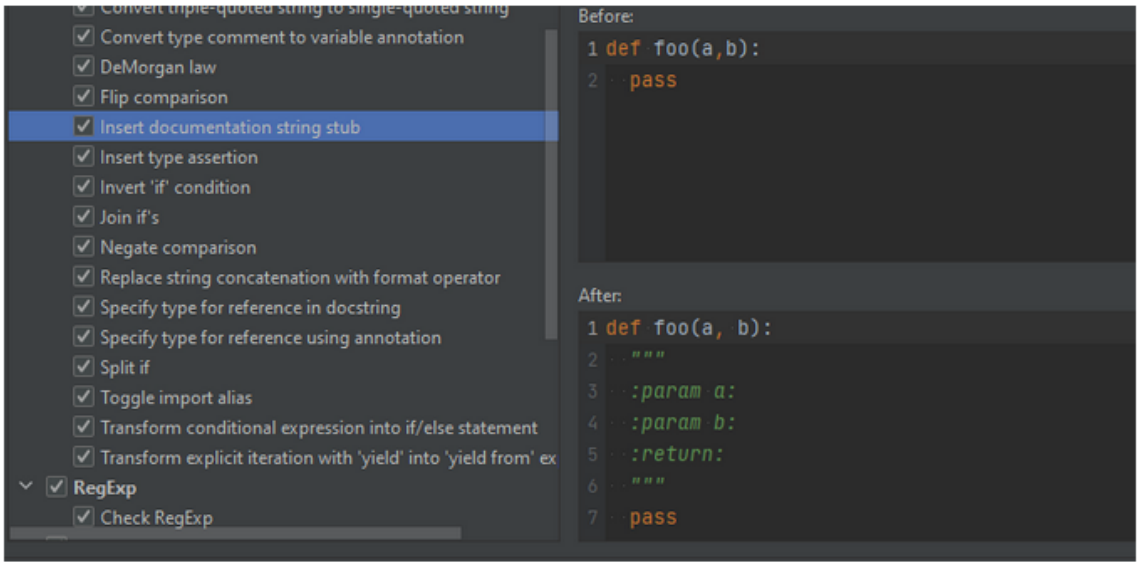
Para ello, vamos a *File -> Settings -> Tools -> Pyhton Integrated Tools.*



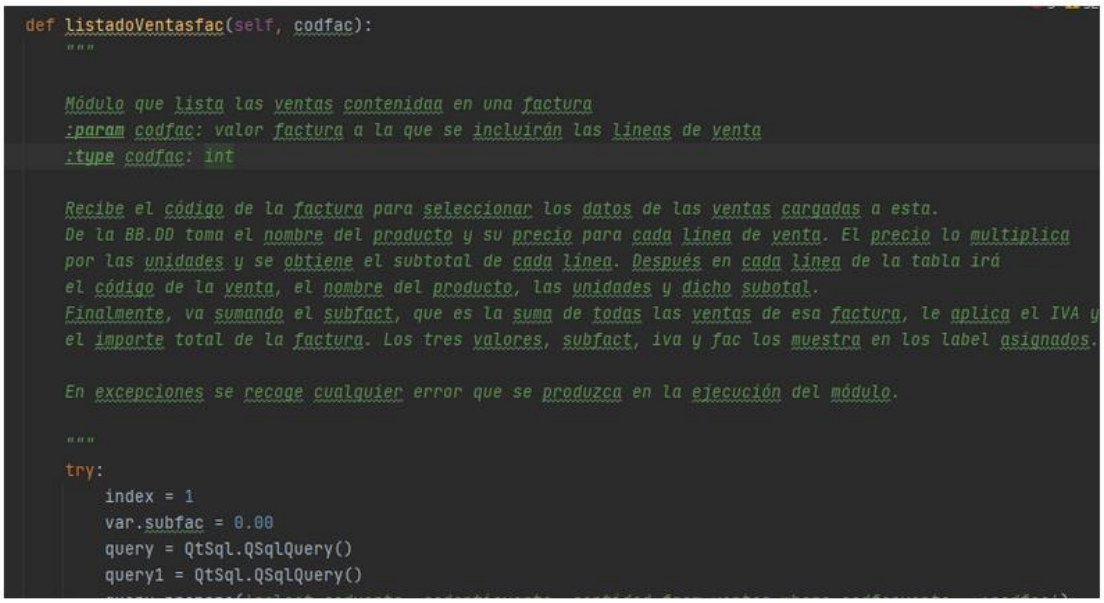
Con las herramientas listas, vamos a un módulo que queramos documentar, nos situamos en el nombre y clickeamos en **Insert Documentation string stub**.

IMPORTANTE.- Es necesario en los módulos que reciben un parámetro que aparezca acompañado por el **self,** por ejemplo, ***AltaCli(self, newCliente),*** en caso contrario Sphinx no muestra el valor ***:param*** (ver imagen)






```
def listadoVentasfac(self, codfac):
    """

    :param codfac:
    :type codfac:
    """
    try:
        index = 1
        var.subfac = 0.00
        query = QtSql.QSqlQuery()
        query1 = QtSql.QSqlQuery()
        query.prepare('select codventa, codarticventa, cantidad from ven
        query.bindValue(':codfac', int(codfac))
        if query.exec_():
```



```
def listadoVentasfac(self, codfac):
    """

    Módulo que lista las ventas contenidaa en una factura
    :param codfac: valor factura a la que se incluirán las lineas de venta
    :type codfac: int

    Recibe el código de la factura para seleccionar los datos de las ventas cargadas a esta.
    De la BB.DD toma el nombre del producto y su precio para cada linea de venta. El precio lo multiplica
    por las unidades y se obtiene el subtotal de cada linea. Despues en cada linea de la tabla irá
    el código de la venta, el nombre del producto, las unidades y dicho subotal.
    Finalmente, va sumando el subfact, que es la suma de todas las ventas de esa factura, le aplica el IVA y
    el importe total de la factura. Los tres valores, subfact, iva y fac los muestra en los label asignados.

    En excepciones se recoge cualquier error que se produzca en la ejecución del módulo.

    """
    try:
        index = 1
        var.subfac = 0.00
        query = QtSql.QSqlQuery()
        query1 = QtSql.QSqlQuery()
```

En Python, **un *docstring*** o **cadena de documentación** es una cadena de caracteres que se coloca como **primer enunciado de un módulo.**

- La **primera línea** de la cadena de documentación debe ser una línea de **resumen terminada con un punto.**

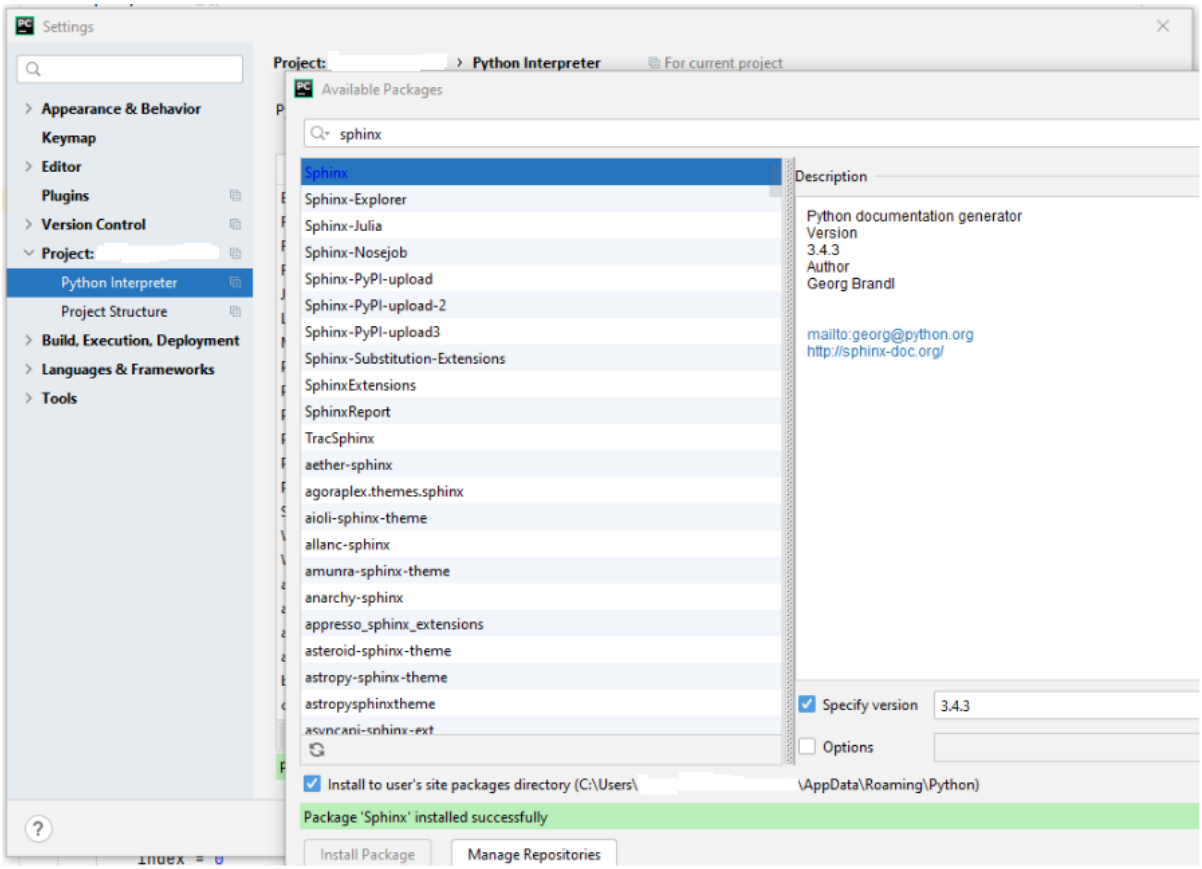
- **El resto** de la documentación **debe describir el comportamiento de la función**, los **valores que devuelve**, las **excepciones que arroja.**

- Se recomienda **dejar una línea en blanco antes de las triples comillas** que cierran la cadena de documentación.

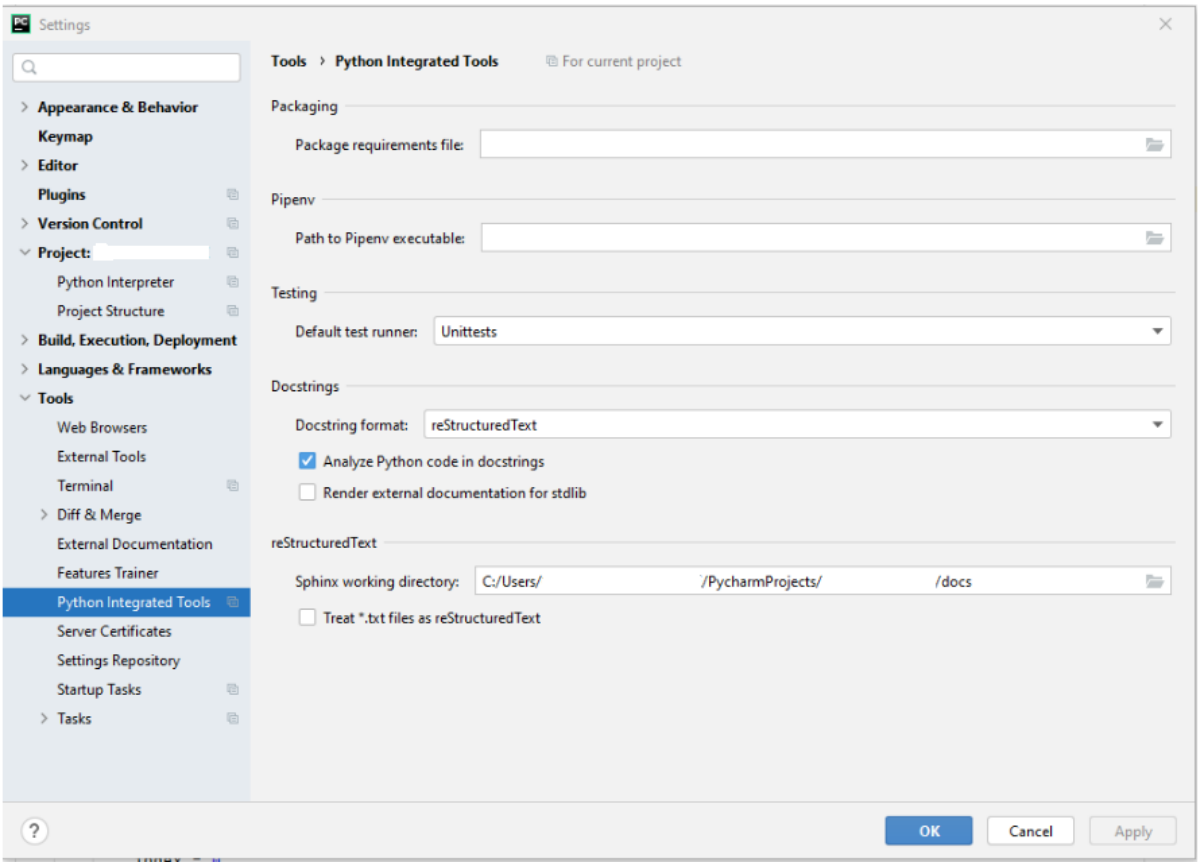
En el siguiente **enlace** se muestran los criterios generales para establecer la sintaxis aconsejable en la documentación de los ***docstrings.***

El último paso es la generación de la documentación. Para ello enumeraremos una a una las fases:

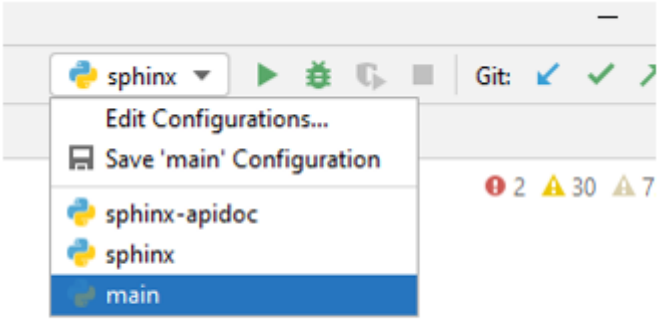
1. Instalación de ***Sphinx.***



2. Nos vamos a *File -> Settings -> Tools -> Python Integrated Tools* y configuramos lo siguiente, **sin olvidarnos crear la carpeta *docs* en el proyecto.**

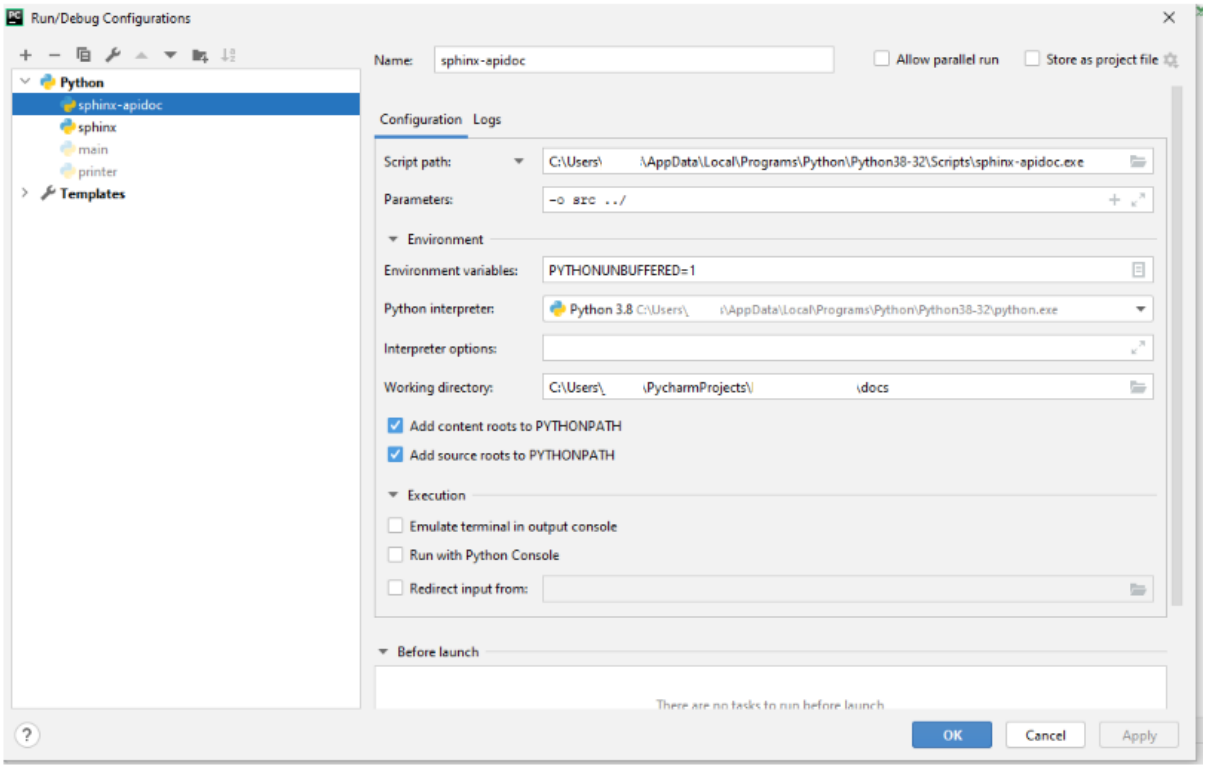


3. En *Run/Debug Configurations*

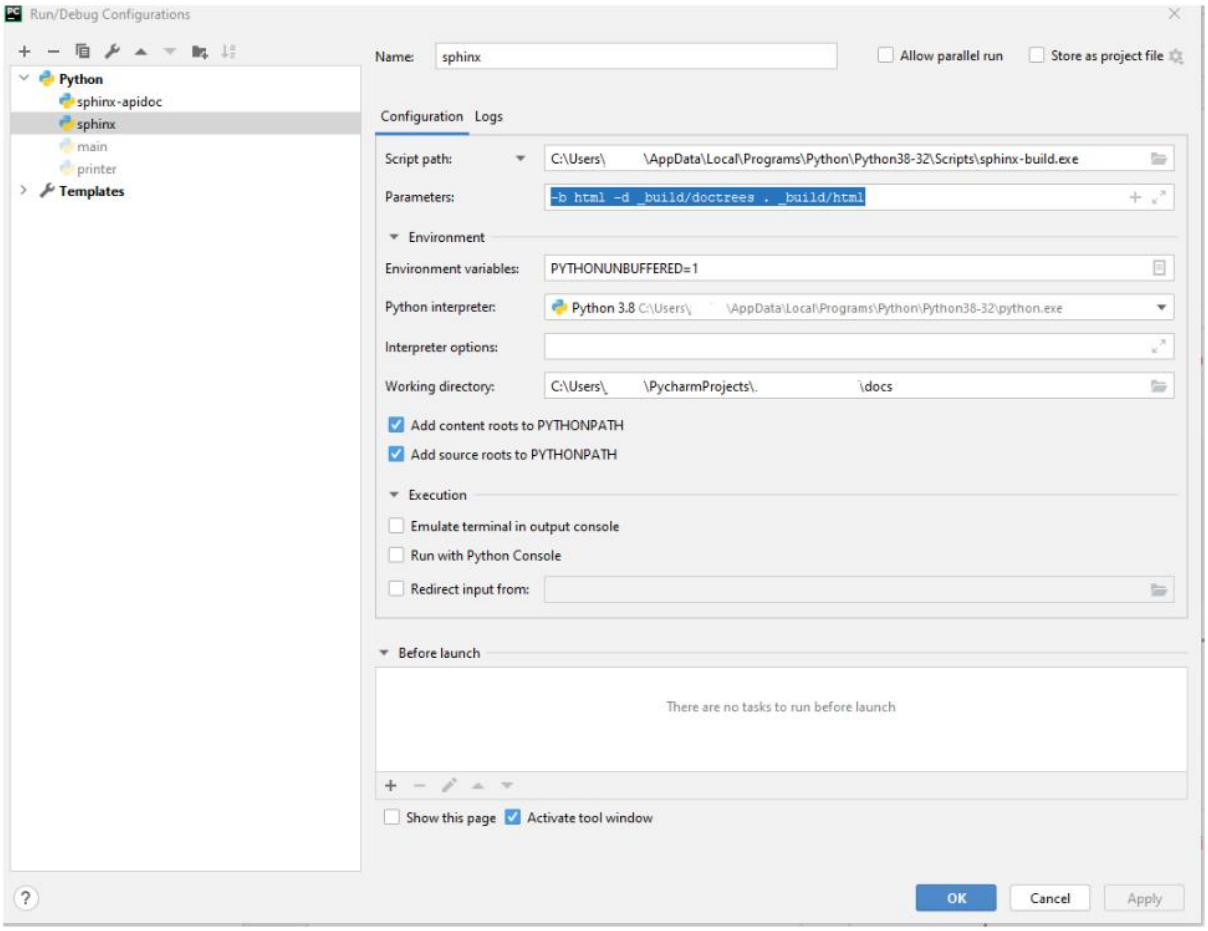


configuramos lo siguiente:

A) La herramienta ***sphinx-apidoc*** que genera la configuración de la documentación

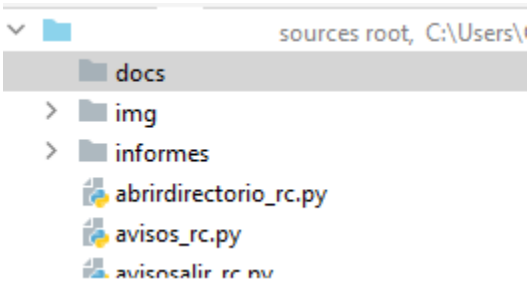


B) La herramienta ***sphinx*** que genera la documentación propiamente dicha.



Ya tenemos listas las herramientas para generar la documentación en html. A continuación, comienza el propio proceso de documentación del programa.

En primer lugar, creamos el directorio **docs.**



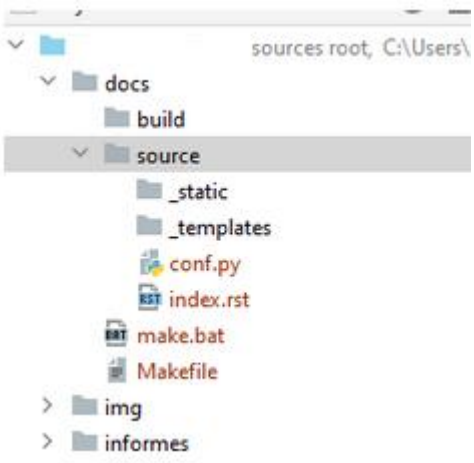
A continuación se ejecuta en ***Tools-Sphinx-Quicstart*** para crear los ficheros de configuración y la estructura de la documentación (no usar tildes).

En la siguiente imagen se muestra un ejemplo de la documentación resultante de la ejecución de las herramientas.

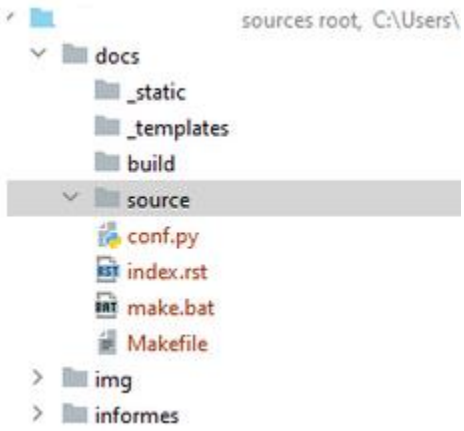
```
C:\Users\                    \AppData\Local\Programs\Python\Python39\python.exe "C:\Program Files\JetBrains\PyCharm Community Edition 2020.2.
Welcome to the Sphinx 3.4.3 quickstart utility.

Please enter values for the following settings (just press Enter to
accept a default value, if one is given in brackets).

Selected root path: .

You have two options for placing the build directory for Sphinx output.
Either, you use a directory "_build" within the root path, or you separate
"source" and "build" directories within the root path.
> Separate source and build directories (y/n) [n]: y

The project name will occur in several places in the built documentation.
> Project name: PROYECTODAM20_21
> Author name(s):
> Project release []: v.0.0.1

If the documents are to be written in a language other than English,
you can select a language here by its language code. Sphinx will then
translate text that it generates into that language.

For a list of supported codes, see
https://www.sphinx-doc.org/en/master/usage/configuration.html#confval-language.
> Project language [en]: es

Creating file C:\Users\                    \PycharmProjects\            \docs\source\conf.py.
Creating file C:\Users\                    \PycharmProjects\            \docs\source\index.rst.
Creating file C:\Users\                    \PycharmProjects\            \docs\Makefile.
Creating file C:\Users\                    \PycharmProjects\            \docs\make.bat.

Finished: An initial directory structure has been created.

You should now populate your master file C:\Users\                    \PycharmProjects\            \docs\source\index.rst and create other d
source files. Use the Makefile to build the docs, like so:
   make builder
where "builder" is one of the supported builders, e.g. html, latex or linkcheck.


Process finished with exit code 0
```

Recordar **mover todo a *docs*** tal como se muestra en la figura.

Antes:



Después:

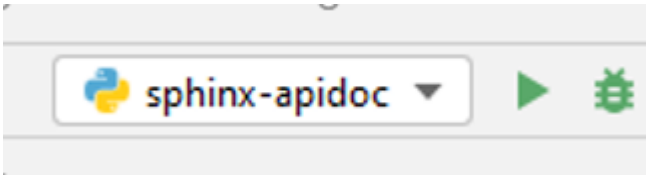


Ahora toca **configurar algunos elementos de *conf.py*** (se muestra en dos imágenes).

```
# -- Path setup --------------------------------------------------------------

# If extensions (or modules to document with autodoc) are in another directory,
# add these directories to sys.path here. If the directory is relative to the
# documentation root, use os.path.abspath to make it absolute, like shown here.
#
import os
import sys
sys.path.insert(0, os.path.abspath('C:\\Users\\       \\PycharmProjects\\                    '))

# Si no funciona: sys.path.insert(0, os.path.abspath('C:\Users\       \PycharmProjects\                    '))
# sin las dobles barras si el usuario empieza por c


# -- Project information -----------------------------------------------------

project = '                  '
copyright = '                  '
author = '           '

# The full version, including alpha/beta/rc tags
release = '0.0.1'


# -- General configuration ---------------------------------------------------

# Add any Sphinx extension module names here, as strings. They can be
# extensions coming with Sphinx (named 'sphinx.ext.*') or your custom
# ones.
extensions = ['sphinx.ext.autodoc',
    'sphinx.ext.intersphinx',
    'sphinx.ext.ifconfig',
    'sphinx.ext.viewcode',
    'sphinx.ext.githubpages',
]

# Add any paths that contain templates here, relative to this directory.
templates_path = ['_templates']
```

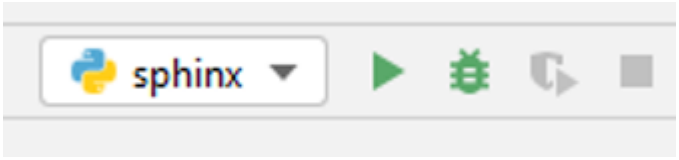
```
# Add any paths that contain templates here, relative to this directory.
templates_path = ['_templates']

# The language for content autogenerated by Sphinx. Refer to documentation
# for a list of supported languages.
#
# This is also used if you do content translation via gettext catalogs.
# Usually you set "language" from the command line for these cases.
language = 'es'

# List of patterns, relative to source directory, that match files and
# directories to ignore when looking for source files.
# This pattern also affects html_static_path and html_extra_path.
exclude_patterns = []

source_suffix = '.rst'


# -- Options for HTML output -------------------------------------------------

# The theme to use for HTML and HTML Help pages.  See the documentation for
# a list of builtin themes.
# https://www.sphinx-doc.org/en/master/usage/theming.html
html_theme = 'classic'


# Add any paths that contain custom static files (such as style sheets) here,
# relative to this directory. They are copied after the builtin static files,
# so a file named "default.css" will overwrite the builtin "default.css".
# html_static_path = ['_static']
```

Ahora ejecutamos ***Sphinx-apidoc.*** Puede mostar algún ***warning*** *que ignoraremos.*



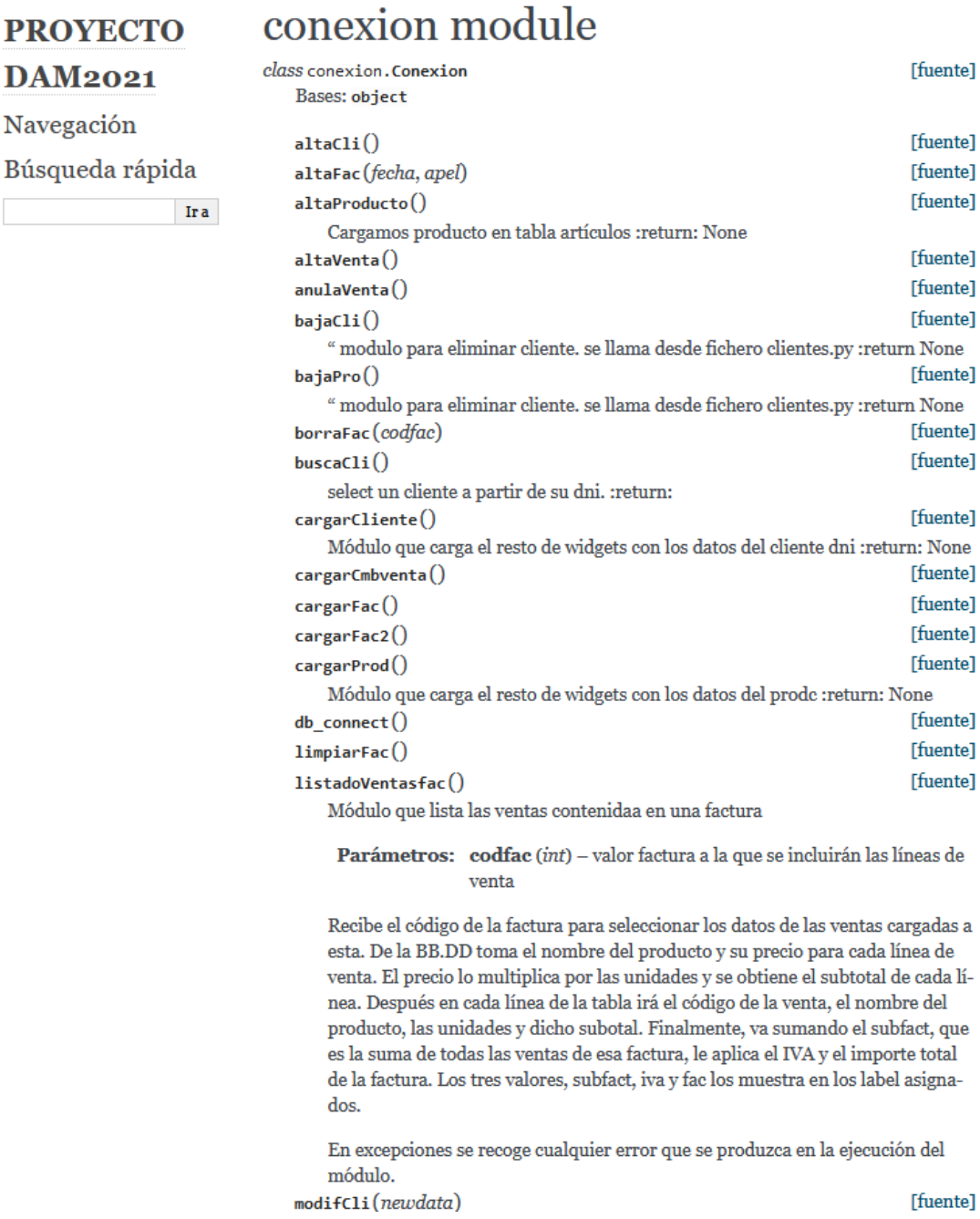
```
C:\Users\                    \AppData\Local\Programs\Python\Python39\python.exe C:/Users/                    /AppD
Creating file src\abrirdirectorio_rc.rst.
Creating file src\avisos_rc.rst.
Creating file src\avisosalir_rc.rst.
Creating file src\clients.rst.
Creating file src\conexion.rst.
Creating file src\events.rst.
Creating file src\logo_rc.rst.
Creating file src\main.rst.
Creating file src\printer.rst.
Creating file src\products.rst.
Creating file src\toolbarbackup_rc.rst.
Creating file src\toolbarprinter_rc.rst.
Creating file src\toolbarsalir_rc.rst.
Creating file src\var.rst.
Creating file src\venavisos.rst.
Creating file src\vencalendar.rst.
Creating file src\vensalir.rst.
Creating file src\ventana.rst.
Creating file src\ventas.rst.
Creating file src\modules.rst.

Process finished with exit code 0
```

Y finalmente **sphinx,**




```
sphinx
C:\Users\                    \AppData\Local\Programs\Python\Python39\python.exe C:/Users/                    /AppData/Roaming/Py
Running Sphinx v3.4.3
loading translations [es]... done
WARNING: Support for evaluating Python 2 syntax is deprecated and will be removed in Sphinx 4.0. Convert C:\Users\
loading pickled environment... done
building [mo]: targets for 0 po files that are out of date
building [html]: targets for 1 source files that are out of date
updating environment: 0 added, 1 changed, 0 removed
reading sources... [100%] index
looking for now-outdated files... none found
pickling environment... done
checking consistency... done
preparing documents... C:\Users\                    \PycharmProjects\                \docs\src\modules.rst: WARNING: document isn't
done

generating indices... genindex py-modindex done
highlighting module code... [100%] ventas
writing additional pages... search done
copying static files... done
copying extra files... done
dumping search index in Spanish (code: es)... done
dumping object inventory... done
build succeeded, 2 warnings.

The HTML pages are in _build\html.
```

Si vamos a ***_build\html*** y abrimos *index.html* nos muestra la documentación generada, en este caso del módulo conexión.



PROYECTO DAM2021

Navegación

Búsqueda rápida

Ira

conexion module

class conexion.**Conexion** [fuente]

Bases: **object**

altaCli() [fuente]

altaFac(*fecha*, *apel*) [fuente]

altaProducto() [fuente]

Cargamos producto en tabla artículos :return: None

altaVenta() [fuente]

anulaVenta() [fuente]

bajaCli() [fuente]

“ modulo para eliminar cliente. se llama desde fichero clientes.py :return None

bajaPro() [fuente]

“ modulo para eliminar cliente. se llama desde fichero clientes.py :return None

borraFac(*codfac*) [fuente]

buscaCli() [fuente]

select un cliente a partir de su dni. :return:

cargarCliente() [fuente]

Módulo que carga el resto de widgets con los datos del cliente dni :return: None

cargarCmbventa() [fuente]

cargarFac() [fuente]

cargarFac2() [fuente]

cargarProd() [fuente]

Módulo que carga el resto de widgets con los datos del prodc :return: None

db_connect() [fuente]

limpiarFac() [fuente]

listadoVentasfac() [fuente]

Módulo que lista las ventas contenidaa en una factura

Parámetros: **codfac** (*int*) – valor factura a la que se incluirán las líneas de venta

Recibe el código de la factura para seleccionar los datos de las ventas cargadas a esta. De la BB.DD toma el nombre del producto y su precio para cada línea de venta. El precio lo multiplica por las unidades y se obtiene el subtotal de cada línea. Después en cada línea de la tabla irá el código de la venta, el nombre del producto, las unidades y dicho subotal. Finalmente, va sumando el subfact, que es la suma de todas las ventas de esa factura, le aplica el IVA y el importe total de la factura. Los tres valores, subfact, iva y fac los muestra en los label asignados.

En excepciones se recoge cualquier error que se produzca en la ejecución del módulo.

modifCli(*newdata*) [fuente]

Más información:

https://stackoverflow.com/questions/46341816/how-to-generate-html-documentation-with-python-sphinx