Manifest: __manifest__.py

The manifest file serves to declare a python package as an Odoo module and to specify module metadata.

It is a file called `__manifest__.py` and contains a single Python dictionary, where each key specifies module metadatum.

```
{
    'name': "A Module",
    'version': '1.0',
    'depends': ['base'],
    'author': "Author Name",
    'category': 'Category',
    'description': """
    Description text
    """,
    # data files always loaded at installation
    'data': [
        'views/mymodule_view.xml',
    ],
    # data files containing optionally loaded demonstration data
    'demo': [
        'demo/demo_data.xml',
    ],
}
```

Available manifest fields are:

`name` (`str`, required) - the human-readable name of the module

`version` (`str`) - this module's version, 1.0, 1.0.1

`description` (`str`) - extended description for the module, in reStructuredText

`author` (`str`) - name of the module author

`website` (`str`) - website URL for the module author

`license` (`str`, defaults: `LGPL-3`) - distribution license for the module. Possible values:

- `GPL-2`
- `GPL-2 or any later version`
- `GPL-3`

- `GPL-3 or any later version`
- `AGPL-3`
- `LGPL-3`
- `Other OSI approved licence`
- `OEEL-1` (Odoo Enterprise Edition License v1.0)
- `OPL-1` (Odoo Proprietary License v1.0)
- `Other proprietary`

`category` (`str`, default: `Uncategorized`) - classification category within Odoo, rough business domain for the module.

`depends` (`list(str)`) - Odoo modules which must be loaded before this one, either because this module uses features they create or because it alters resources they define.

`data` (`list(str)`) - List of data files which must always be installed or updated with the module. A list of paths from the module root directory

`demo` (`list(str)`) - List of data files which are only installed or updated in *demonstration mode*

`auto_install` (`bool` or `list(str)`, default: `False`) - If `True`, this module will automatically be installed if all of its dependencies are installed.

`external_dependencies` (`dict(key=list(str))`) - A dictionary containing python and/or binary dependencies.

```
{
    'Python': ['idap']
}
```

`application` (`bool`, default: `False`) - Whether the module should be considered as a fully-fledged application (`True`) or is just a technical module (`False`) that provides some extra functionality to an existing application module.

`assets` (`dict`) - A definition of how all static files are loaded in various assets bundles.

`installable` (`bool` default: `True`) - Whether a user should be able to install the module from the Web UI or not.

`maintainer` (`str`) - Person or entity in charge of the maintenance of this module, by default it is assumed that the author is the maintainer.

`{pre_init, post_init, uninstall}_hook` (`str`)

Hooks for module installation/uninstallation, their value should be a string representing the name of a function defined inside the module's `__init__.py`.

`pre_init_hook` - takes a cursor as its only argument, this function is executed prior to the module's installation.

`post_init_hook` - takes a cursor and a registry as its arguments, this function is executed right after the module's installation.

`uninstall_hook` - takes a cursor and a registry as its arguments, this function is executed after the module's uninstallation.

These hooks should only be used when setup/cleanup required for this module is either extremely difficult or impossible through the api.