iradinaGUI Documentation

Release 1.0.0

CEA DEN/DANS/DM2S/STMF/LGLS

Oct 06, 2020

CONTENTS

Warning:

1. Find a *pdf* version of this documentation here[1].
2. Find *Iradina code* manual here[2].

The iradinaGUI code is a GUI[3] (Graphical User Interface) used to perform operations with Iradina code.

This GUI code is a set of Python3[4] scripts files.

[1] iradinaGUI/doc/build/latex/iradinaGUI.pdf
[2] iradinaGUI/doc/src/iradinaDocuments/20140804_iradina_manual.pdf
[3] https://en.wikipedia.org/wiki/Graphical_user_interface
[4] https://docs.python.org/3.5

CHAPTER

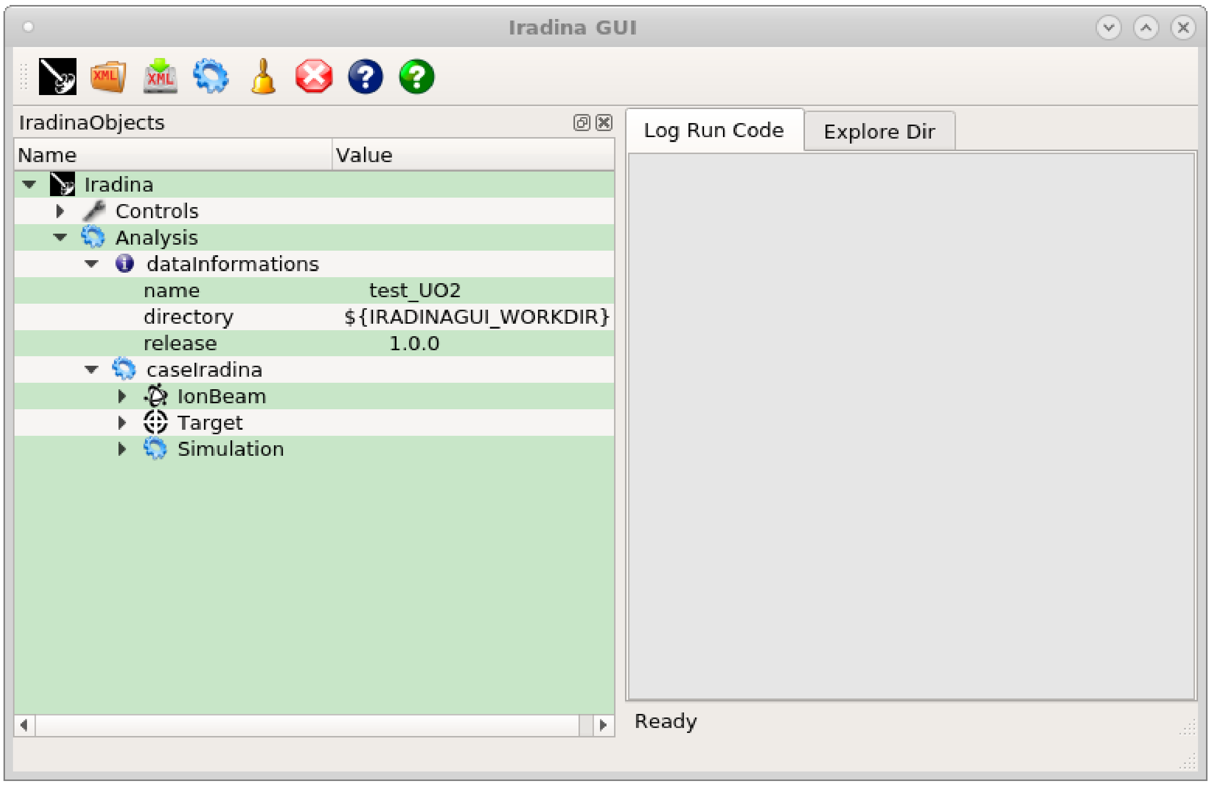
ONE

USER'S MANUAL

1.1 Iradina GUI main widget

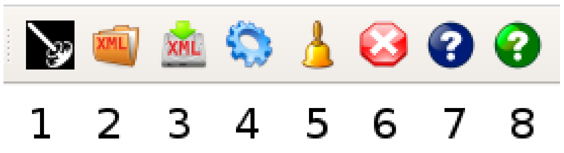
From this main widget named *Iradina GUI* users can:

1. Prepare Iradina code data.
2. Run Iradina code.
3. Plot curves from result files of Iradina code runs.



Name	Value
Iradina	
Controls	
Analysis	
dataInformations	
name	test_UO2
directory	${IRADINAGUI_WORKDIR}
release	1.0.0
caseIradina	
IonBeam	
Target	
Simulation	

1.1.1 Main widget toolbar



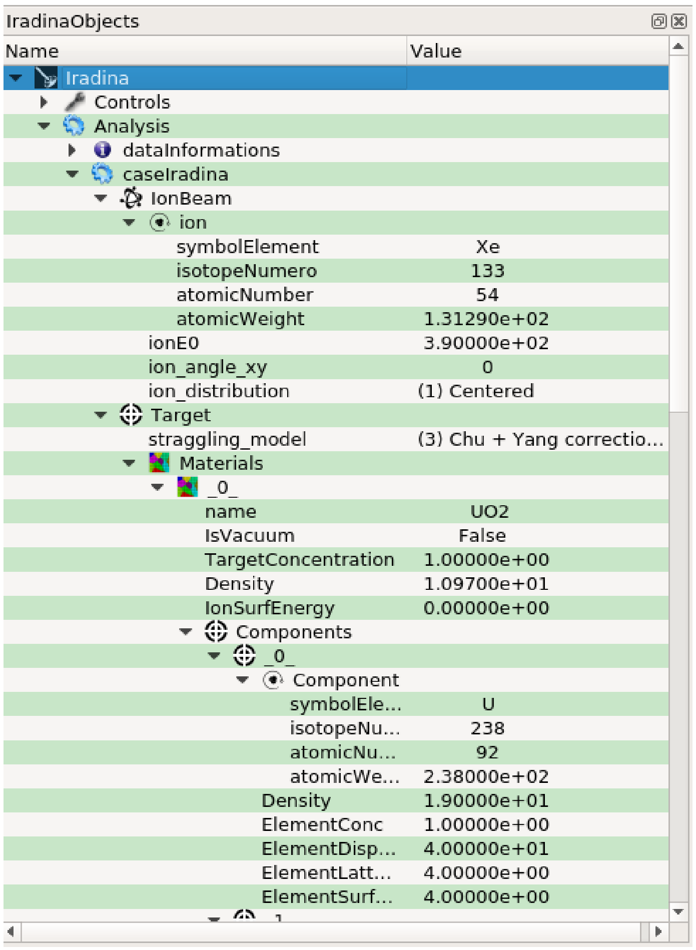
This toolbar contains icons related to actions, from left to right:

1. New Iradina data. Create iradina case from scratch
2. Load Iradina data. Load case from previously saved case in a file Xml.
3. Save Iradina data. Save current iradina case in a file Xml.
4. Launch Iradina calculus. Launch Iradina code on current iradina case.
5. Refresh IradinaObjects tree view.
6. Clear Iradina data model, remove Iradina data tree (in IradinaObject widget).
7. Iradina GUI help. Display this current documentation in a browser (html mode).
8. Iradina code help. Display Iradina code manual in a browser (pdf mode).

1.2 Iradina tree view widget

From this tree view widget named *IradinaObjects* users can:

1. Prepare Iradina code data.



IradinaObjects

Name	Value
Iradina	
Controls	
Analysis	
dataInformations	
caseIradina	
IonBeam	
ion	
symbolElement	Xe
isotopeNumero	133
atomicNumber	54
atomicWeight	1.31290e+02
ionE0	3.90000e+02
ion_angle_xy	0
ion_distribution	(1) Centered
Target	
straggling_model	(3) Chu + Yang correctio...
Materials	
_0_	
name	UO2
IsVacuum	False
TargetConcentration	1.00000e+00
Density	1.09700e+01
IonSurfEnergy	0.00000e+00
Components	
_0_	
Component	
symbolEle...	U
isotopeNu...	238
atomicNu...	92
atomicWe...	2.38000e+02
Density	1.90000e+01
ElementConc	1.00000e+00
ElementDisp...	4.00000e+01
ElementLatt...	4.00000e+00
ElementSurf...	4.00000e+00

1.2.1 Modify tree view widget items values

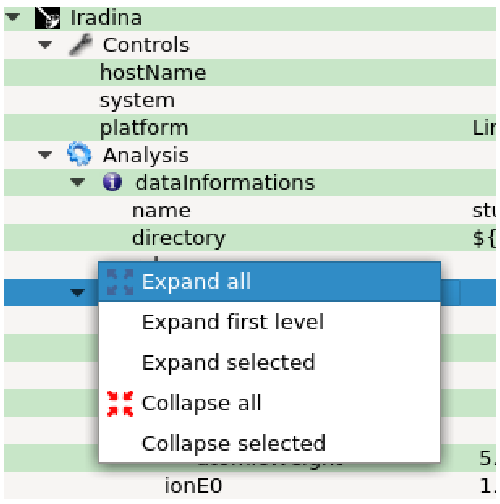
There are some values as leaves of tree. Names and tooltips are *almost* as Iradina code naming usage.

1. **Simple scalar values**. User can modify value on *mouse-left-double-click*, selecting tree item nodes **hovering column value**.
2. **Other specific values**. User can modify value on *mouse-right-click*, to get a contextual menu for modification, selecting tree item nodes **hovering columns name and value**.

1.2.2 Tree view widget menus

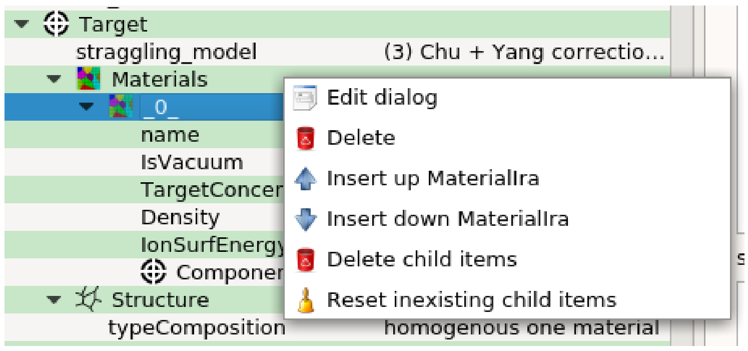
There are some menus, as *contextual menu* on *mouse-right-click* when selected tree item nodes. Some menu are generic, and other are specific to node, as contextual actions. This is a **NOT exhaustive** list of menus:

Expand/collapse menu



This menu contains some actions to expand or collapse all or selected part of data tree. To activate this menu users have to *mouse-right-click* on **head of arrow** of tree item nodes (at **left** of item icon).

Delete/Insert menu

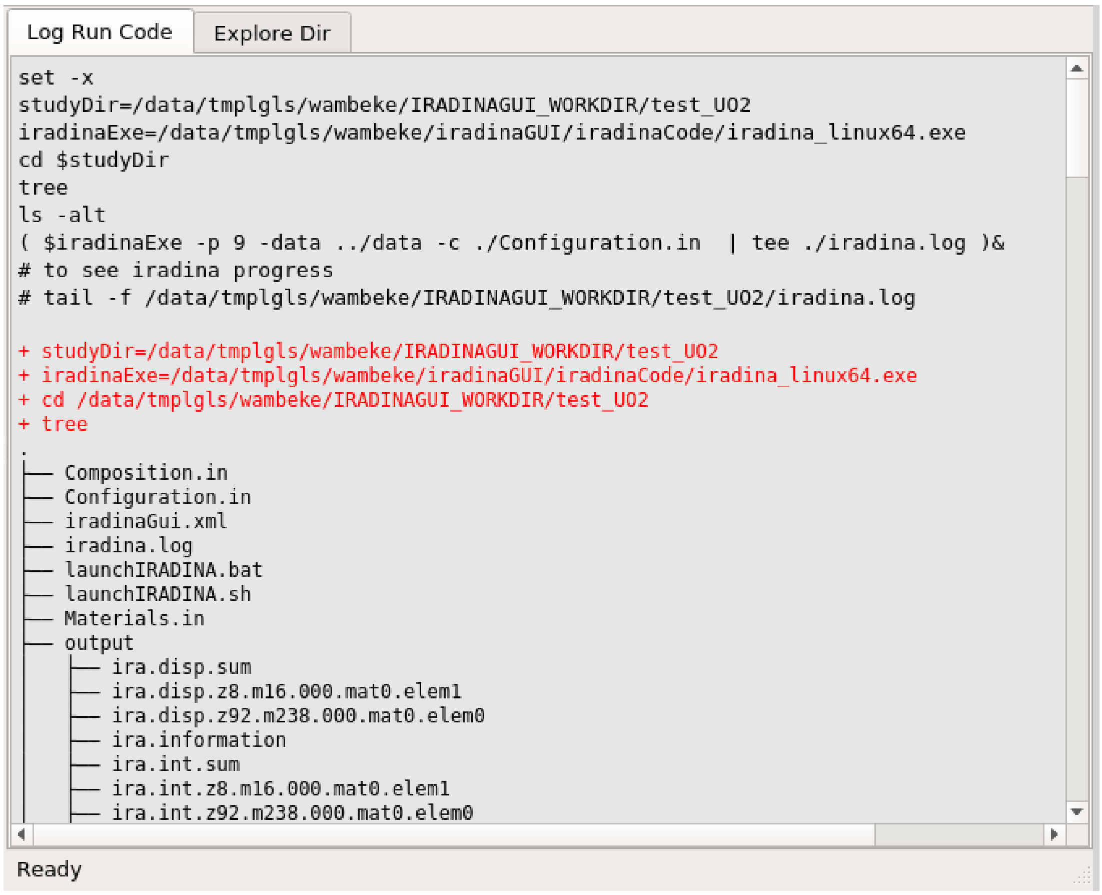


This menu contains some actions to insert, delete and reset all or selected part of data tree. To activate this menu users have to *mouse-right-click* on ‘name’ of tree item nodes (at right of item icon). The concerned items are usually not leaves (are items without a value).

1.3 Log Run Code widget

This widget displays log trace of Iradina code execution.

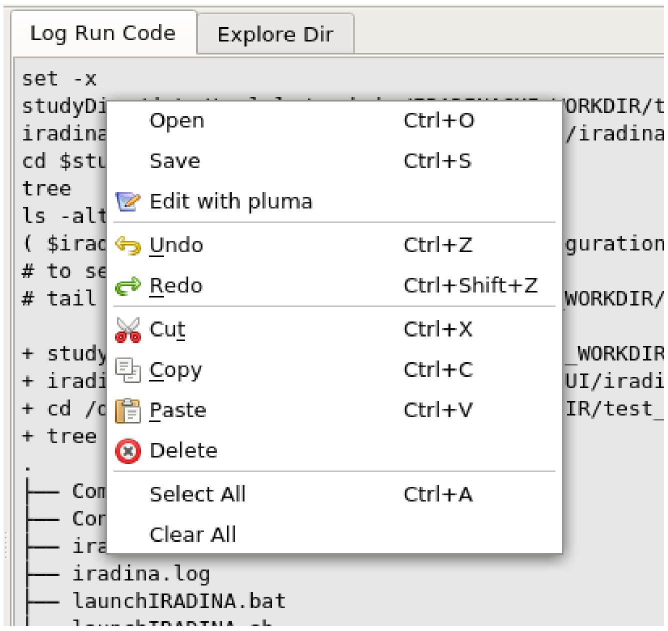
Iradina code is executed when users activate *Launch Iradina calculus* button in *Main widget toolbar* (page 4).



```
set -x
studyDir=/data/tmplgls/wambeke/IRADINAGUI_WORKDIR/test_UO2
iradinaExe=/data/tmplgls/wambeke/iradinaGUI/iradinaCode/iradina_linux64.exe
cd $studyDir
tree
ls -alt
( $iradinaExe -p 9 -data ../data -c ./Configuration.in  | tee ./iradina.log )&
# to see iradina progress
# tail -f /data/tmplgls/wambeke/IRADINAGUI_WORKDIR/test_UO2/iradina.log

+ studyDir=/data/tmplgls/wambeke/IRADINAGUI_WORKDIR/test_UO2
+ iradinaExe=/data/tmplgls/wambeke/iradinaGUI/iradinaCode/iradina_linux64.exe
+ cd /data/tmplgls/wambeke/IRADINAGUI_WORKDIR/test_UO2
+ tree
.
├── Composition.in
├── Configuration.in
├── iradinaGui.xml
├── iradina.log
├── launchIRADINA.bat
├── launchIRADINA.sh
├── Materials.in
├── output
│   ├── ira.disp.sum
│   ├── ira.disp.z8.m16.000.mat0.elem1
│   ├── ira.disp.z92.m238.000.mat0.elem0
│   ├── ira.information
│   ├── ira.int.sum
│   ├── ira.int.z8.m16.000.mat0.elem1
│   ├── ira.int.z92.m238.000.mat0.elem0
```

Ready

1.3.1 Log Run Code widget menu



This menu contains some actions to display, but **also** edit all or selected part of current log trace, considering log trace as an ascii file. To activate this menu users have to *mouse-right-click* **somewhere in** Log Run Code widget.

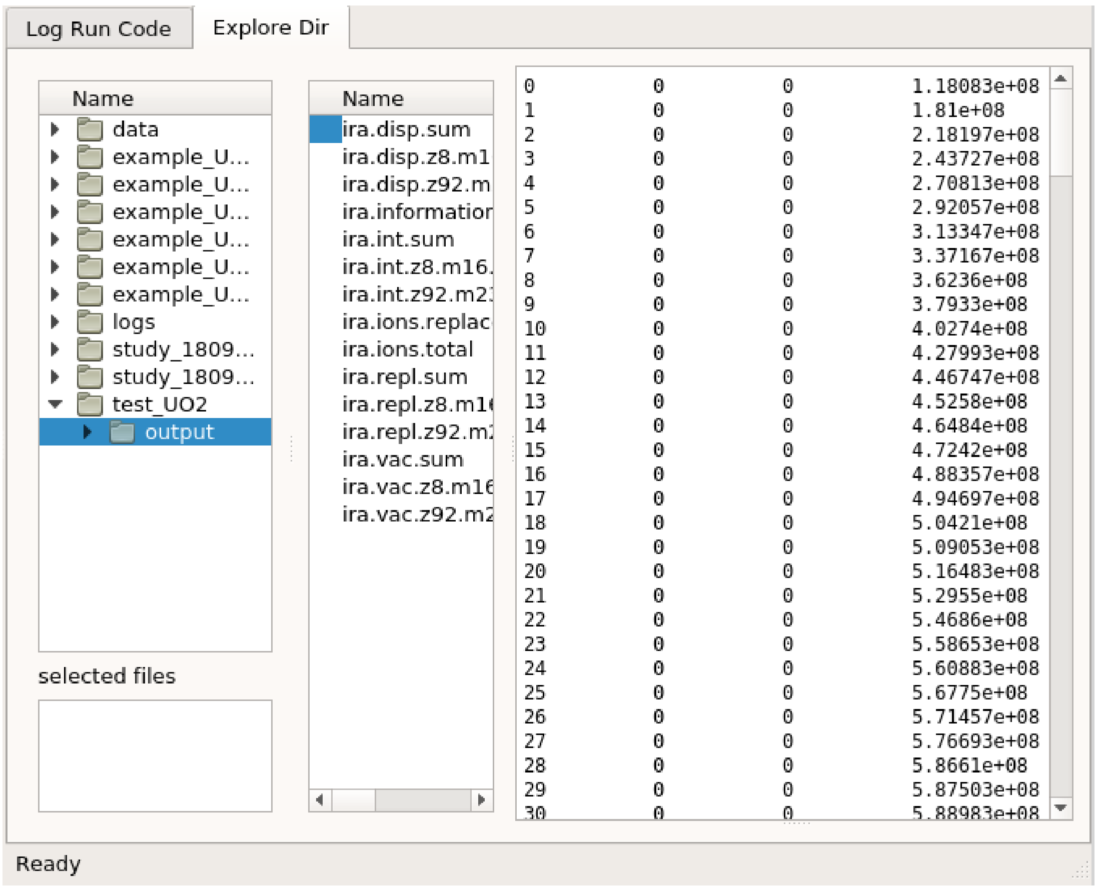
Note: Using *Open* and *Save* actions in this menu, users can use this widget as an elementary text file viewer/editor.

1.4 Explore Dir widget

This widget displays the contents of user iradinaGUI working directory. This directory is usually referenced as *IRADINAGUI_WORKDIR*.

Its usage is like a **simple** file explorer.

Note: Theses result files of Iradina code are located in sub-directories named *output*.



```
data
example_U...
example_U...
example_U...
example_U...
example_U...
example_U...
logs
study_1809...
study_1809...
test_UO2
  output
```

```
ira.disp.sum
ira.disp.z8.m1
ira.disp.z92.m
ira.informatior
ira.int.sum
ira.int.z8.m16.
ira.int.z92.m2
ira.ions.replac
ira.ions.total
ira.repl.sum
ira.repl.z8.m1
ira.repl.z92.m
ira.vac.sum
ira.vac.z8.m16
ira.vac.z92.m2
```

```
0     0     0     1.18083e+08
1     0     0     1.81e+08
2     0     0     2.18197e+08
3     0     0     2.43727e+08
4     0     0     2.70813e+08
5     0     0     2.92057e+08
6     0     0     3.13347e+08
7     0     0     3.37167e+08
8     0     0     3.6236e+08
9     0     0     3.7933e+08
10    0     0     4.0274e+08
11    0     0     4.27993e+08
12    0     0     4.46747e+08
13    0     0     4.5258e+08
14    0     0     4.6484e+08
15    0     0     4.7242e+08
16    0     0     4.88357e+08
17    0     0     4.94697e+08
18    0     0     5.0421e+08
19    0     0     5.09053e+08
20    0     0     5.16483e+08
21    0     0     5.2955e+08
22    0     0     5.4686e+08
23    0     0     5.58653e+08
24    0     0     5.60883e+08
25    0     0     5.6775e+08
26    0     0     5.71457e+08
27    0     0     5.76693e+08
28    0     0     5.8661e+08
29    0     0     5.87503e+08
30    0     0     5.88983e+08
```

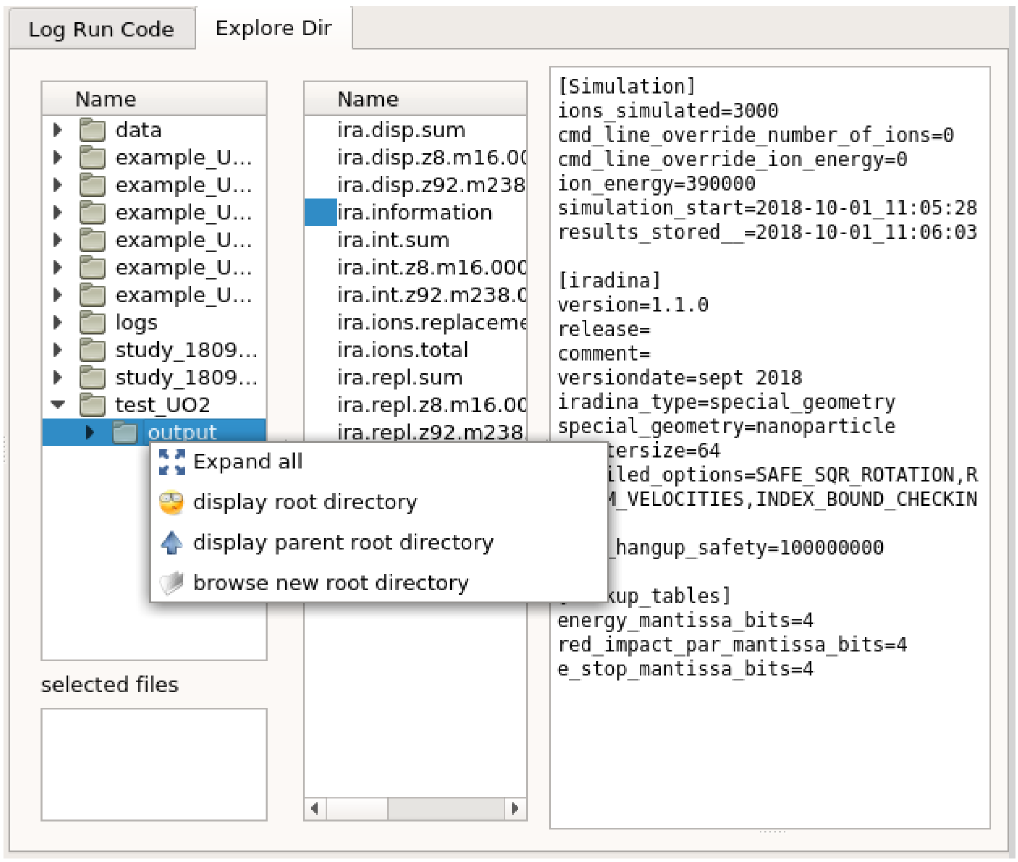
There are three main widgets (from left to right):

1. Directories names widget
2. Files names widget
3. File contents widget

Note: The lower *selected files* widget is for future improvments, no usage *yet*.

There are some contextual menus to explore directories, and to display input/output text files.

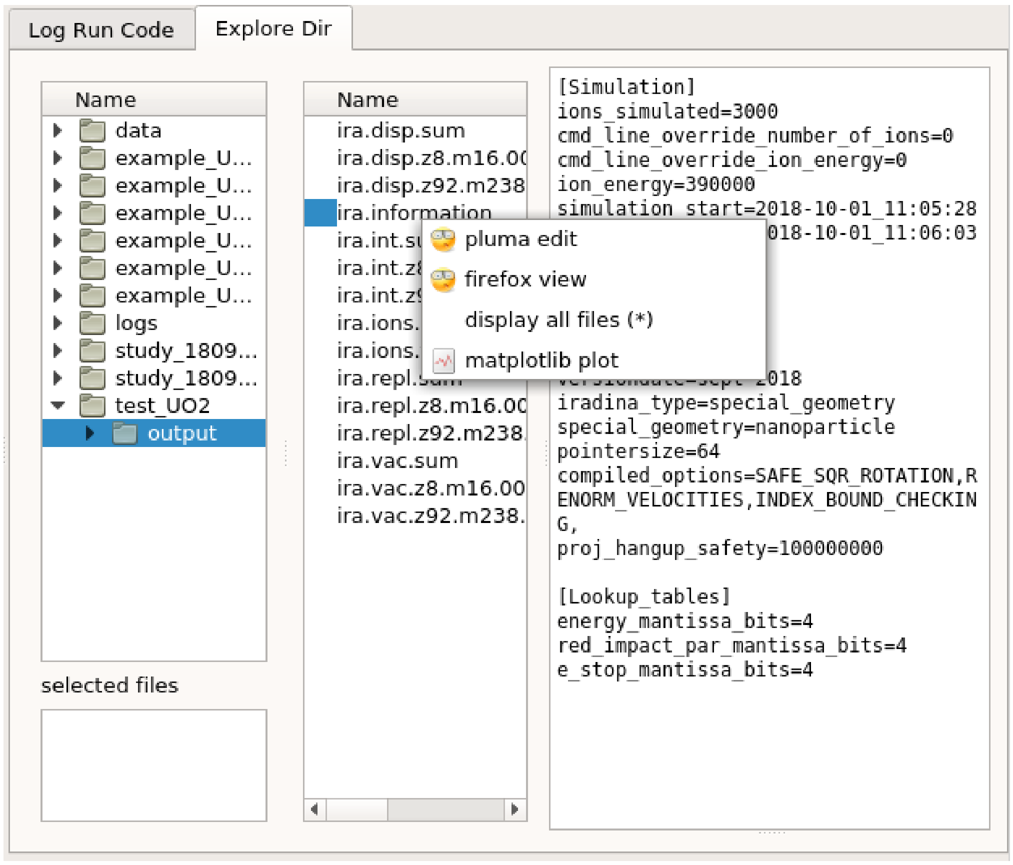
1.4.1 Directories names widget menu



```
[Simulation]
ions_simulated=3000
cmd_line_override_number_of_ions=0
cmd_line_override_ion_energy=0
ion_energy=390000
simulation_start=2018-10-01_11:05:28
results_stored__=2018-10-01_11:06:03

[iradina]
version=1.1.0
release=
comment=
versiondate=sept 2018
iradina_type=special_geometry
special_geometry=nanoparticle
     tersize=64
     iled_options=SAFE_SQR_ROTATION,R
     M_VELOCITIES,INDEX_BOUND_CHECKIN

     _hangup_safety=100000000

     kup_tables]
energy_mantissa_bits=4
red_impact_par_mantissa_bits=4
e_stop_mantissa_bits=4
```

This menu contains some elementary actions to navigate in **all** disk directories.

1.4.2 Files names widget menu

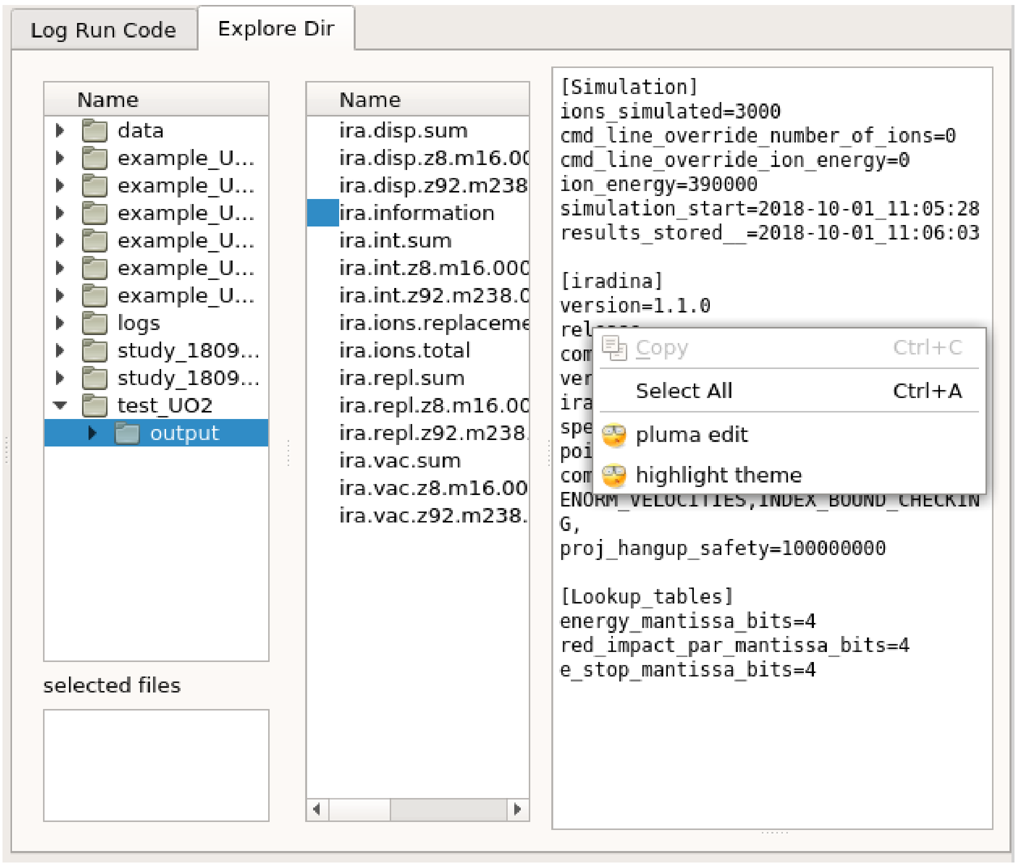


```
[Simulation]
ions_simulated=3000
cmd_line_override_number_of_ions=0
cmd_line_override_ion_energy=0
ion_energy=390000
simulation_start=2018-10-01_11:05:28
                 018-10-01_11:06:03




versiondate=sept 2018
iradina_type=special_geometry
special_geometry=nanoparticle
pointersize=64
compiled_options=SAFE_SQR_ROTATION,R
ENORM_VELOCITIES,INDEX_BOUND_CHECKIN
G,
proj_hangup_safety=100000000

[Lookup_tables]
energy_mantissa_bits=4
red_impact_par_mantissa_bits=4
e_stop_mantissa_bits=4
```

This menu contains some actions to apply on selected file.

Note: An useful action is the lower **matplotlib plot**, to plot curves from result files of Iradina code as post-treatment.

1.4.3 Files contents widget menu



```
[Simulation]
ions_simulated=3000
cmd_line_override_number_of_ions=0
cmd_line_override_ion_energy=0
ion_energy=390000
simulation_start=2018-10-01_11:05:28
results_stored__=2018-10-01_11:06:03

[iradina]
version=1.1.0
rel
com
ver
ira
spe
poi
com
ENORM_VELOCITIES,INDEX_BOUND_CHECKIN
G,
proj_hangup_safety=100000000

[Lookup_tables]
energy_mantissa_bits=4
red_impact_par_mantissa_bits=4
e_stop_mantissa_bits=4
```

This menu contains some elementary actions to apply on displayed file. The files are *syntax highlighted* if possible, using highlight[5] tool, only on Linux distributions for now.

Warning: The lower *highlight theme* action is displayed only for Linux distribution.

[5] http://www.andre-simon.de/doku/highlight/en/highlight.php

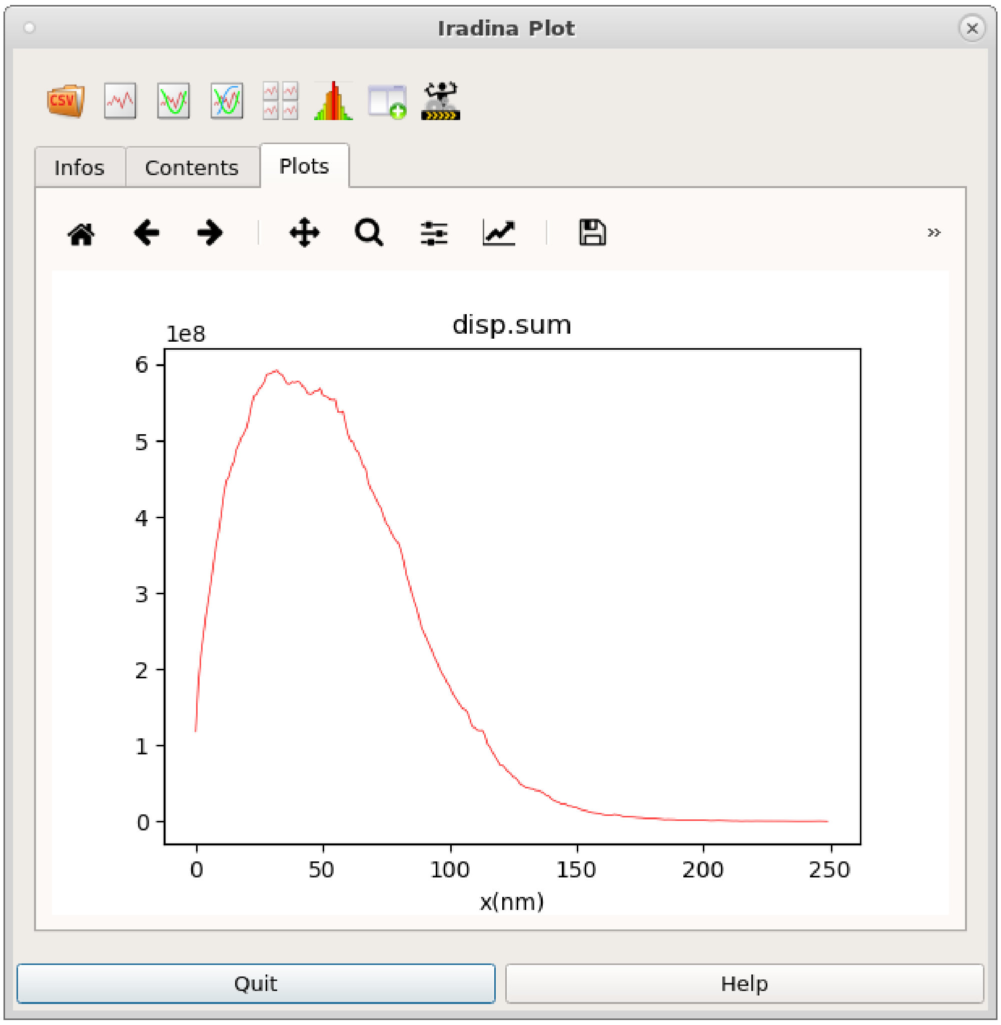
1.5 Plot widget

This widget is to explore, modify and make plots from data. It appears when users activate **matplotlib plot** in *Files names widget menu* (page 11).

It uses some important Python packages:

1. PANDAS[6] (Python Data Analysis Library) to read and store data in memory.
2. MATPLOTLIB[7] (Python 2D plotting library) to plot curves.

Warning: Data are read from CSV[8] formated files. Note that result files of Iradina code use whitespace/tabulation for separator character, *not comma*, for readability.

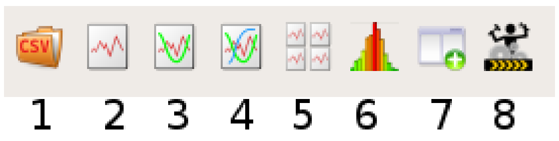


There are four main widgets (from up to down, left to right):

1. Plots toolbar widget
2. Infos tab widget
3. Contents tab widget
4. Plots tab widget

[6] https://pandas.pydata.org
[7] https://matplotlib.org
[8] https://en.wikipedia.org/wiki/Comma-separated_values

1.5.1 Plot toolbar widget



This menu contains some buttons, as actions, which are, from left to right:

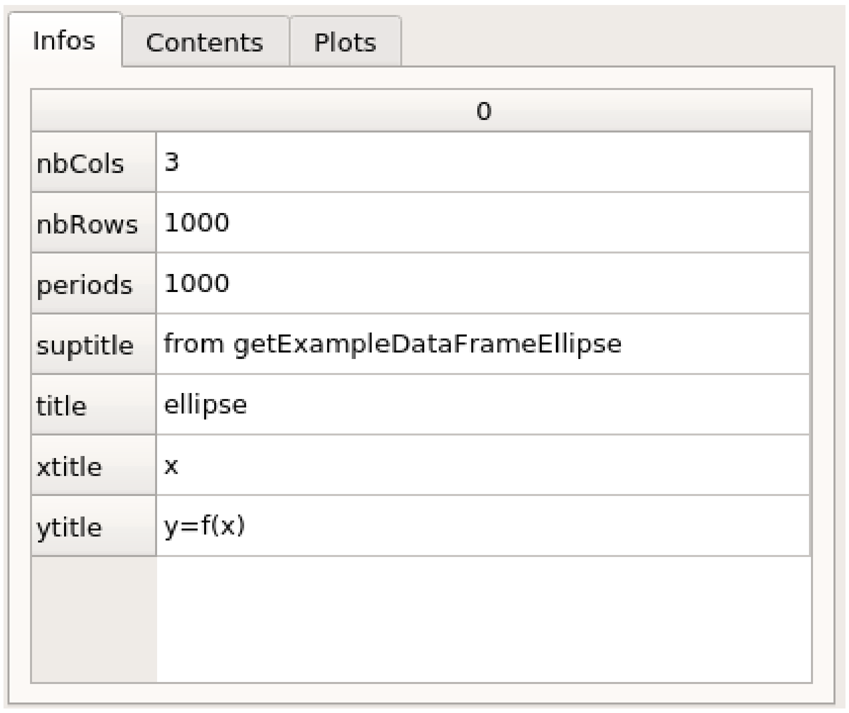
1. Read a new data set from file, CSV[9] formatted.
2. Plot curve y=f(x). 2 columns (x,y) selected.
3. Plot 2 curves (y1,y2)=f(x), two distinct Y-axis. 3 columns (x,y1,y2) selected.
4. Plot n curves y1,. . . yn=f(x), one Y-axis, (n+1) columns (x,y1,. . . yn) selected.
5. Plot n *distinct* curves y1,. . . yn=f(x), (n+1) columns (x,y1,. . . yn) selected.
6. Plot histogram, one columns selected, (automatic number of intervals).
7. Execute PANDAS[10] expressions, modification **on the fly** of *Contents tab widget*.
8. Initialize data **ellipse** in *Contents tab widget* (as a *newbie example*), plot the ellipse.

With this toolbar, user may process:

1. Select one, or more columns in *Contents tab widget*, then call plot of curve(s) using plot buttons of this plot toolbar. A MATPLOTLIB plot is displayed in *Plots tab widget*. See *Example of user plot* (page 18).
2. Create, remove, calculate new(s) colums of data in *Contents tab widget*. See *Example of user PANDAS expression* (page 16).
3. Filter, sort etc. lines of data in *Contents tab widget*, that means also get (and trace) a subset of *Contents tab widget*.

[9] https://en.wikipedia.org/wiki/Comma-separated_values
[10] https://pandas.pydata.org

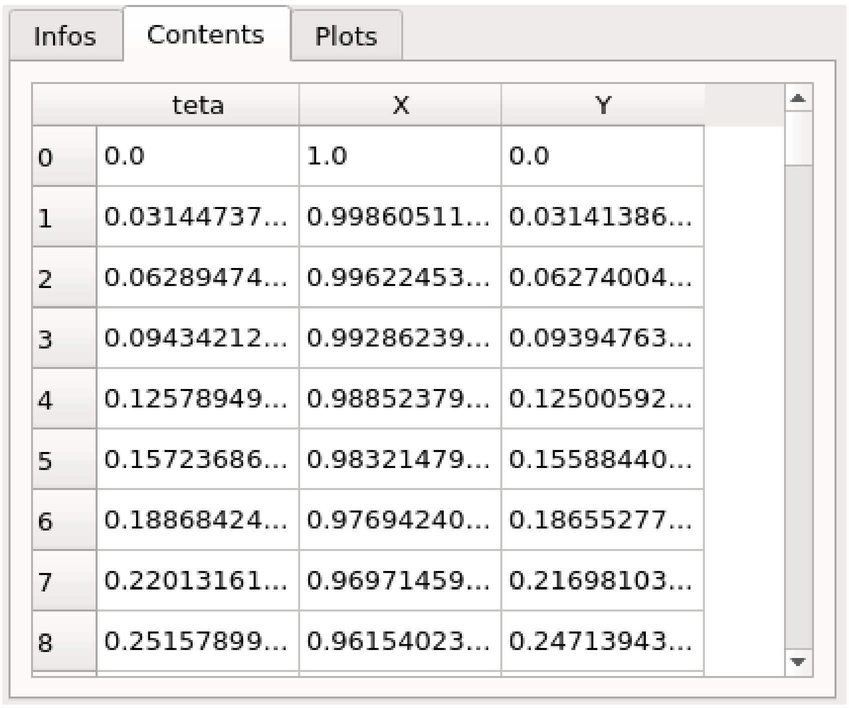
1.5.2 Infos tab widget



	0
nbCols	3
nbRows	1000
periods	1000
suptitle	from getExampleDataFrameEllipse
title	ellipse
xtitle	x
ytitle	y=f(x)

This widget contains some contextual informations from CSV data file, if they are present.

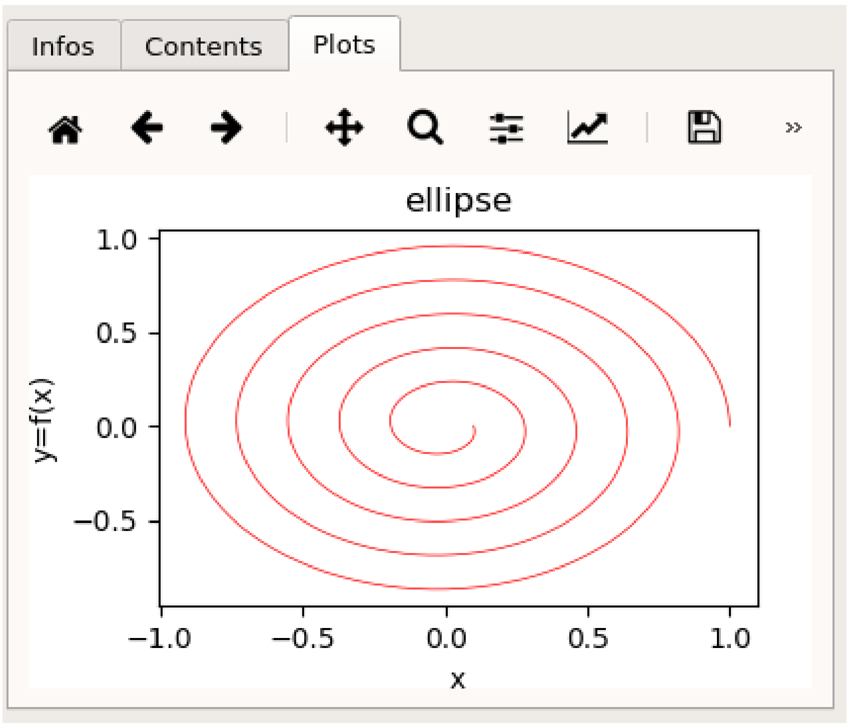
1.5.3 Contents tab widget



	teta	X	Y
0	0.0	1.0	0.0
1	0.03144737...	0.99860511...	0.03141386...
2	0.06289474...	0.99622453...	0.06274004...
3	0.09434212...	0.99286239...	0.09394763...
4	0.12578949...	0.98852379...	0.12500592...
5	0.15723686...	0.98321479...	0.15588440...
6	0.18868424...	0.97694240...	0.18655277...
7	0.22013161...	0.96971459...	0.21698103...
8	0.25157899...	0.96154023...	0.24713943...

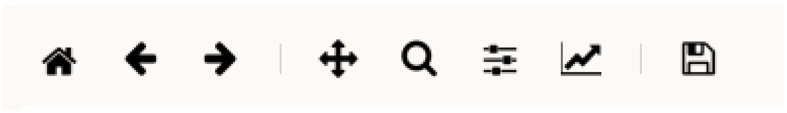
This widget displays current data, allowing some *elementary* modifications (see your appropriate spreadsheet for more functionalities (Excel, or else).

1.5.4 Plots tab widget



This widget displays current plot, as an instance of matplotlib[11] *figure*.

1.5.5 Interactive navigation toolbar matplotlib



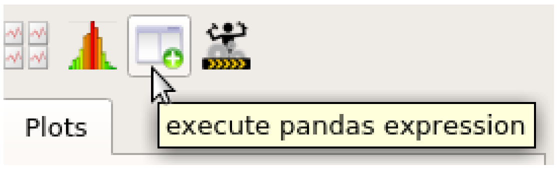
Using this standard toolbar of matplotlib[12], user may customize current plot.

For more informations, users should refer to the description of navigation toolbar MATPLOTLIB[13].

1.6 Plot widget usage

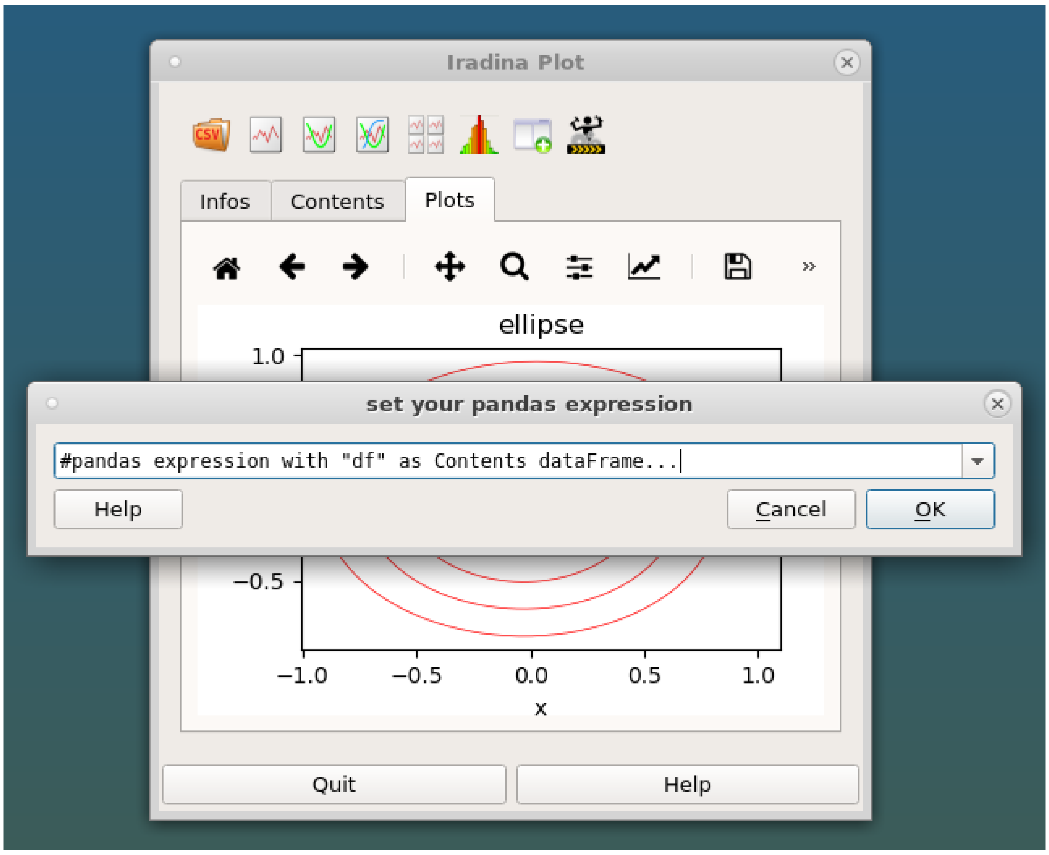
1.6.1 Example of user PANDAS expression

User may modify data of *Contents tab widget* (page 15), Clicking on menu actions button *Execute PANDAS expressions* of *Plot toolbar widget* (page 14).



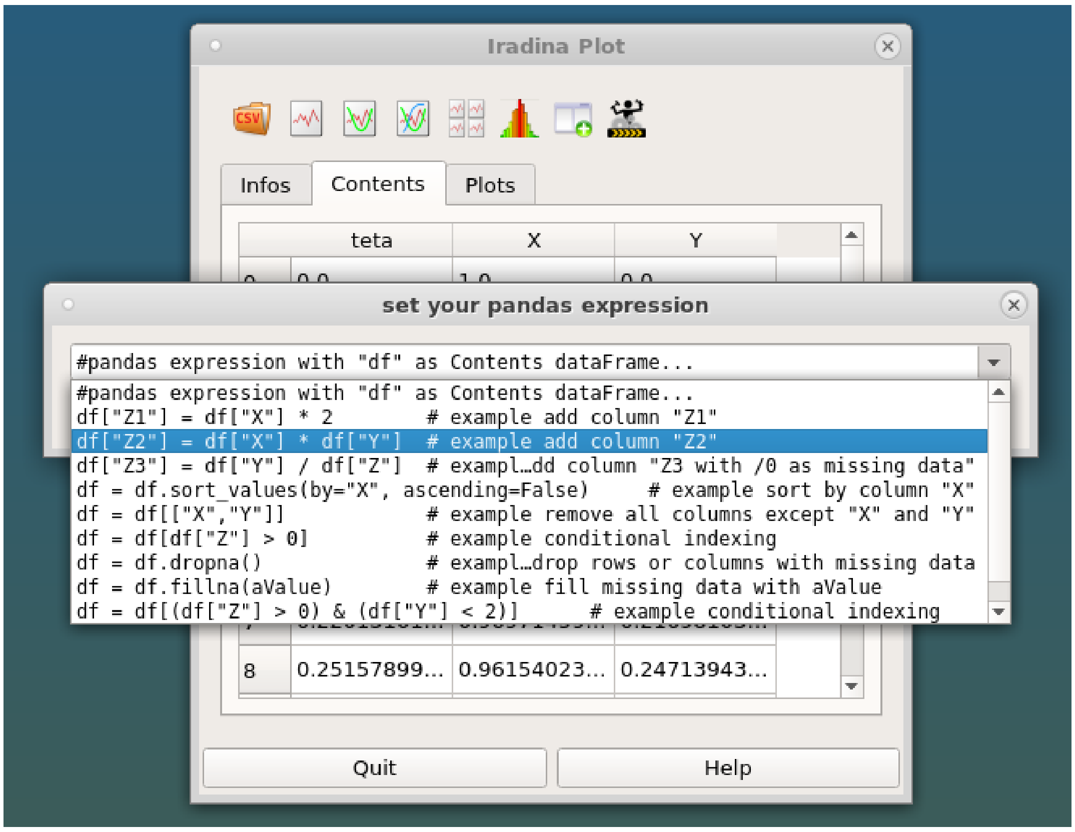
A dialog box appears *set your pandas expression*. An expression is an python affectation expression *'df = something'*, where 'df' *is* a pandas dataframe instance displayed in *Contents tab widget* (page 15).

[11] https://matplotlib.org
[12] https://matplotlib.org
[13] http://matplotlib.org/users/navigation_toolbar.html?highlight=toolbar



```
#pandas expression with "df" as Contents dataFrame...
```

Scrolling, there are some examples of PANDAS[14] dataframe expressions, (read the documentation, sometimes it is **not trivial**).

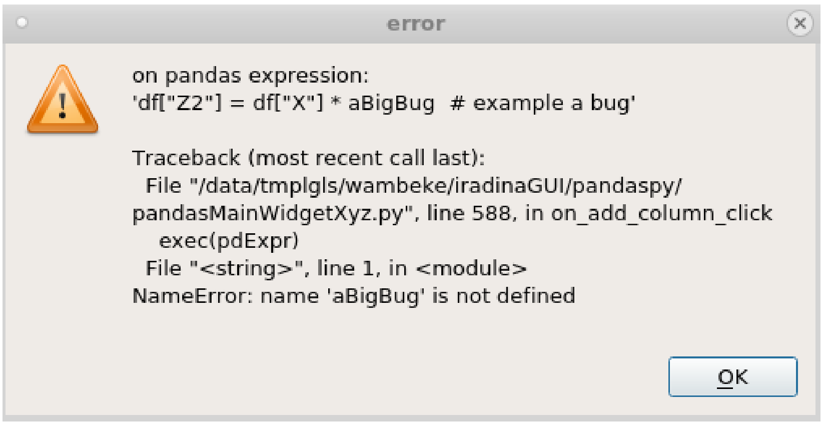


```
#pandas expression with "df" as Contents dataFrame...
#pandas expression with "df" as Contents dataFrame...
df["Z1"] = df["X"] * 2         # example add column "Z1"
df["Z2"] = df["X"] * df["Y"]   # example add column "Z2"
df["Z3"] = df["Y"] / df["Z"]   # exampl…dd column "Z3 with /0 as missing data"
df = df.sort_values(by="X", ascending=False)      # example sort by column "X"
df = df[["X","Y"]]             # example remove all columns except "X" and "Y"
df = df[df["Z"] > 0]           # example conditional indexing
df = df.dropna()               # exampl…drop rows or columns with missing data
df = df.fillna(aValue)         # example fill missing data with aValue
df = df[(df["Z"] > 0) & (df["Y"] < 2)]       # example conditional indexing
```

Users have to type their own expression, and confirm (OK). Do not worry, in case of error(s) in expression, an

[14] https://pandas.pydata.org

error dialog box appears.

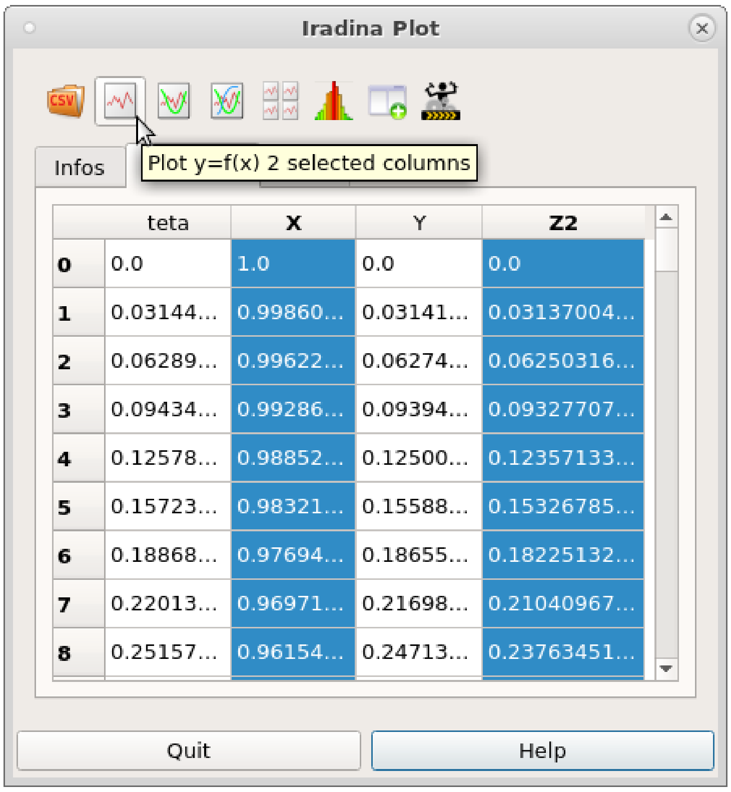


```
on pandas expression:
'df["Z2"] = df["X"] * aBigBug  # example a bug'

Traceback (most recent call last):
  File "/data/tmplgls/wambeke/iradinaGUI/pandaspy/
pandasMainWidgetXyz.py", line 588, in on_add_column_click
    exec(pdExpr)
  File "<string>", line 1, in <module>
NameError: name 'aBigBug' is not defined
```

1.6.2 Example of user plot

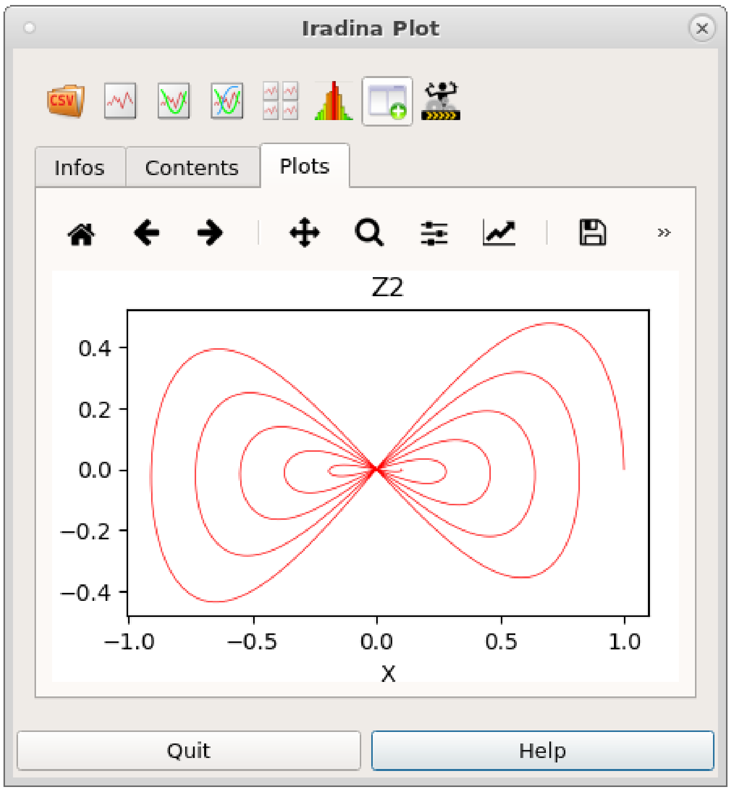
When selecting, *for example*, theses two columns from ellipse: (X, Z2=X*Y), and Clicking on button 2 of *Plot toolbar widget* (page 14).

Warning: Sometimes, users have to make a **multiple-selection** of *expected* number of colums in *Contents tab widget* **before** plot actions. The order of multiple-selection is significant.



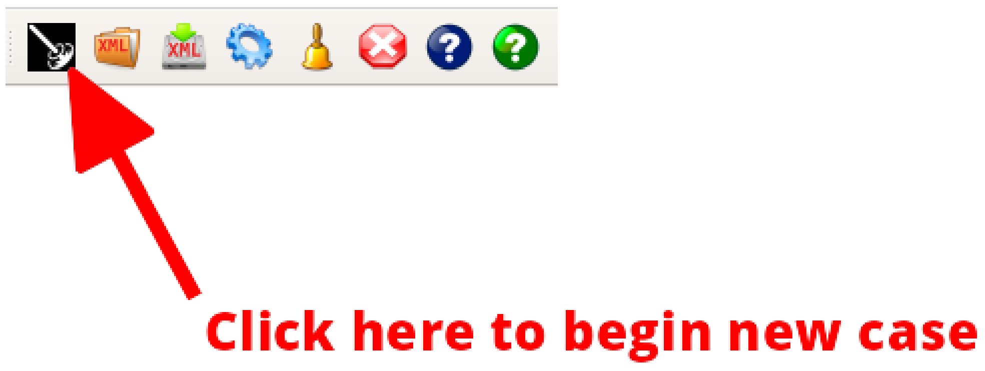
	teta	X	Y	Z2
0	0.0	1.0	0.0	0.0
1	0.03144...	0.99860...	0.03141...	0.03137004...
2	0.06289...	0.99622...	0.06274...	0.06250316...
3	0.09434...	0.99286...	0.09394...	0.09327707...
4	0.12578...	0.98852...	0.12500...	0.12357133...
5	0.15723...	0.98321...	0.15588...	0.15326785...
6	0.18868...	0.97694...	0.18655...	0.18225132...
7	0.22013...	0.96971...	0.21698...	0.21040967...
8	0.25157...	0.96154...	0.24713...	0.23763451...

User get this plot

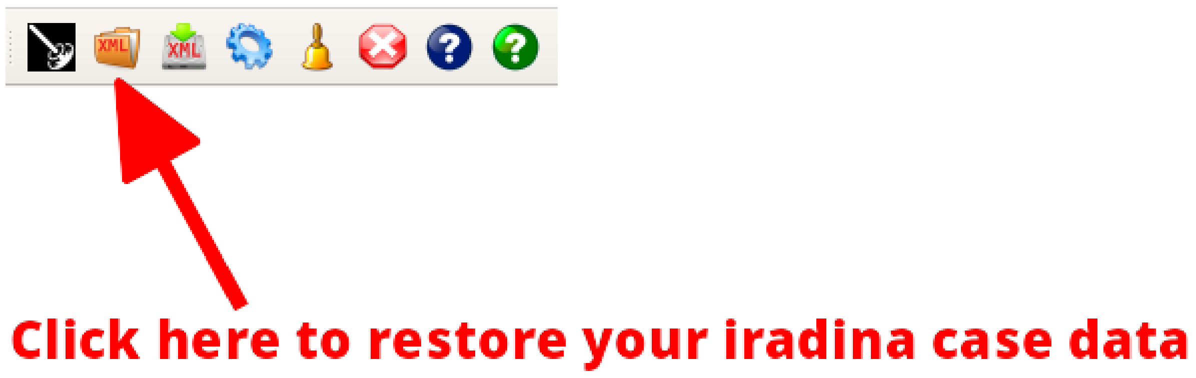

Iradina Plot
Infos
Contents
Plots
Z2
0.4
0.2
0.0
−0.2
−0.4
−1.0
−0.5
0.0
0.5
1.0
X
Quit
Help

1.7 Tips

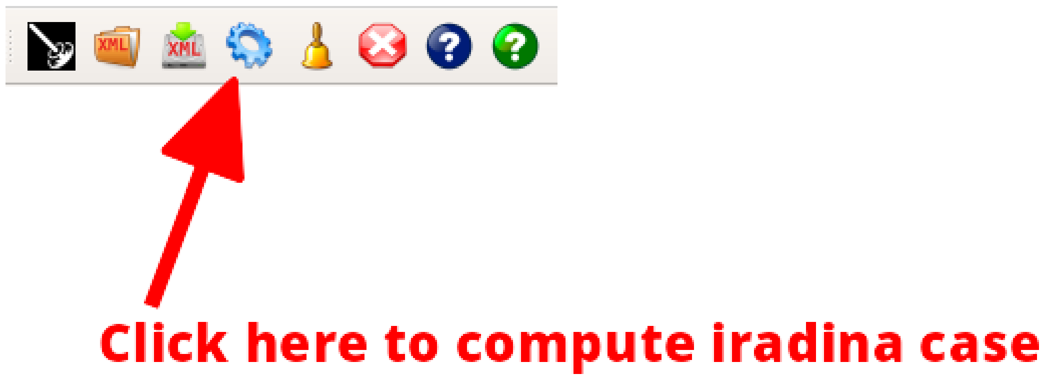
1.7.1 Tip 01



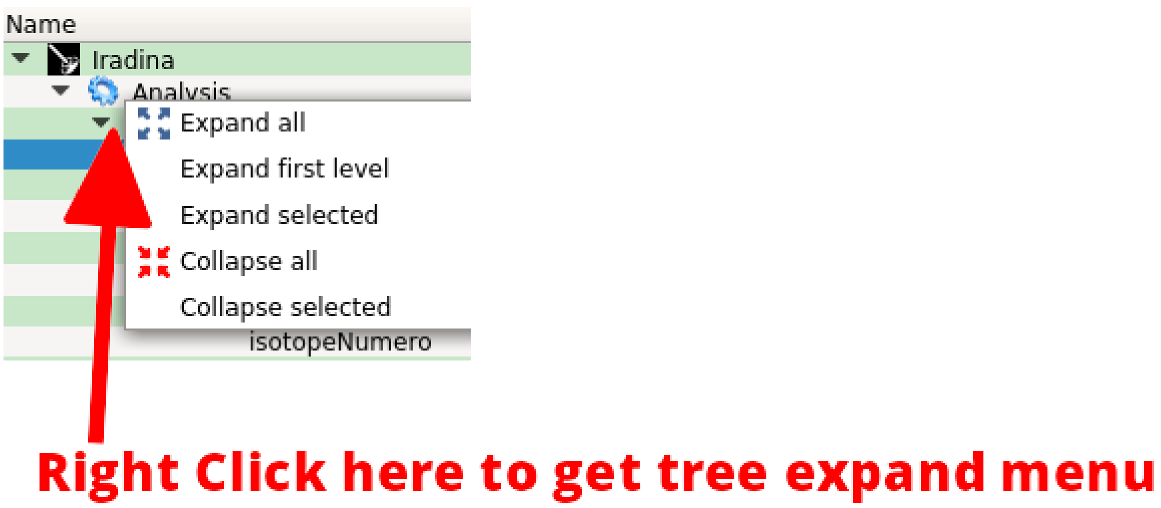
1.7.2 Tip 05



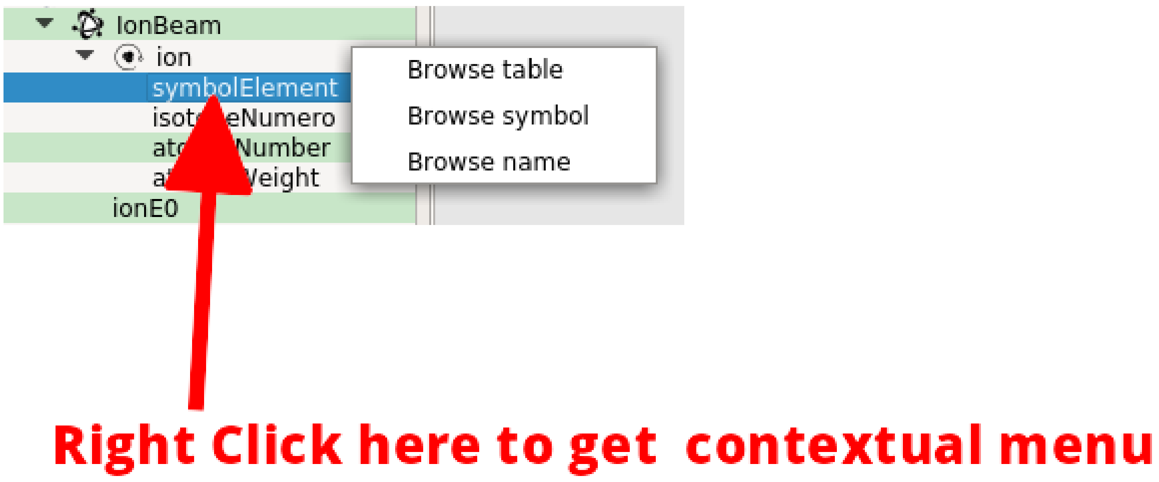
1.7.3 Tip 08



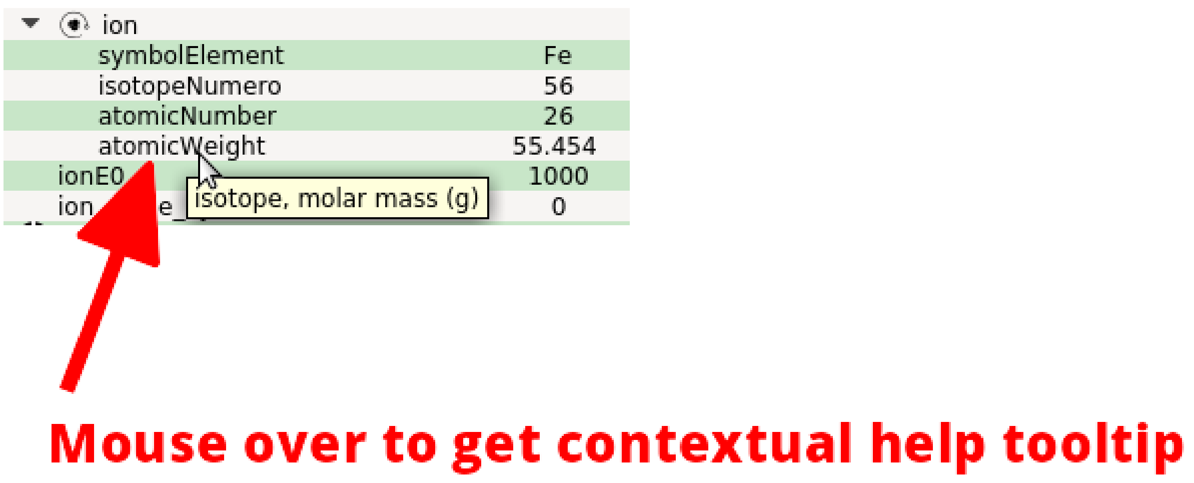
1.7.4 Tip 20



1.7.5 Tip 25



1.7.6 Tip 28

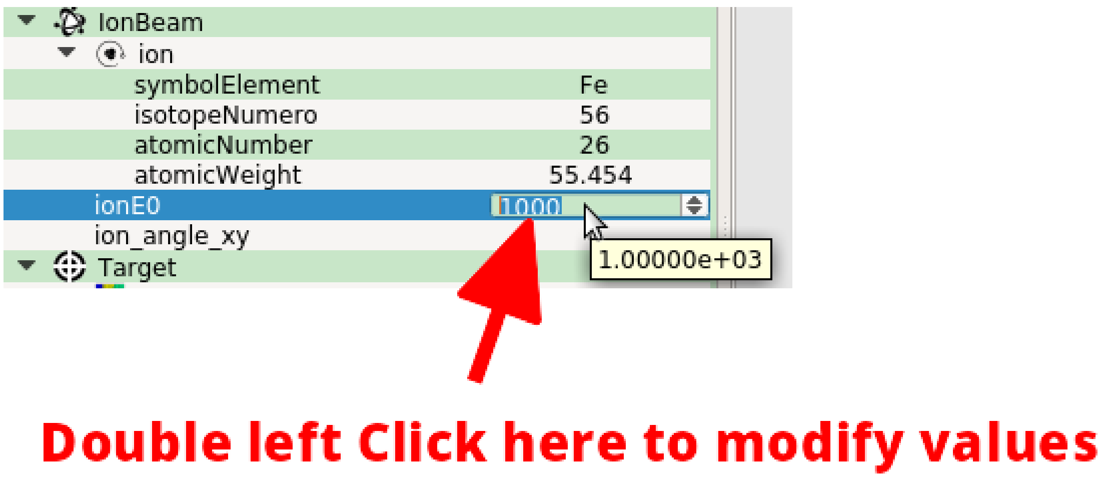


ion	
symbolElement	Fe
isotopeNumero	56
atomicNumber	26
atomicWeight	55.454
ionE0	1000
ion…e_…	0

isotope, molar mass (g)

Mouse over to get contextual help tooltip

1.7.7 Tip 30

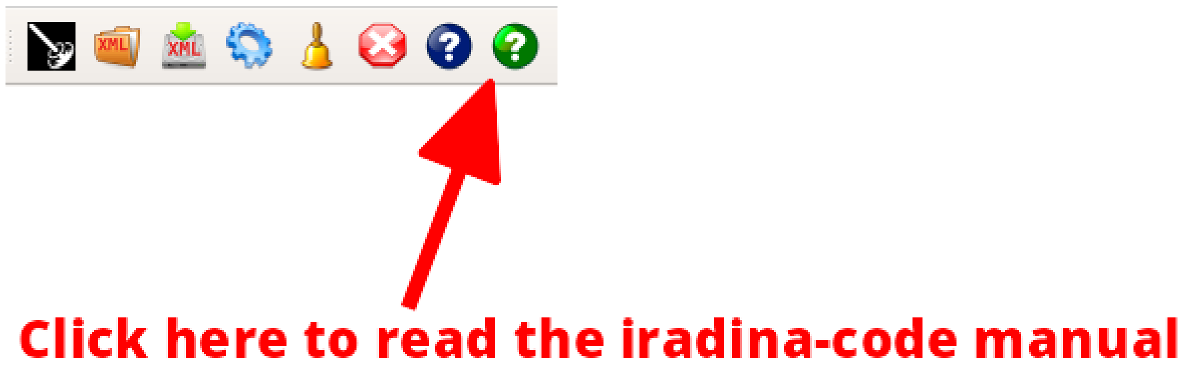


IonBeam	
ion	
symbolElement	Fe
isotopeNumero	56
atomicNumber	26
atomicWeight	55.454
ionE0	1000
ion_angle_xy	
Target	

1.00000e+03

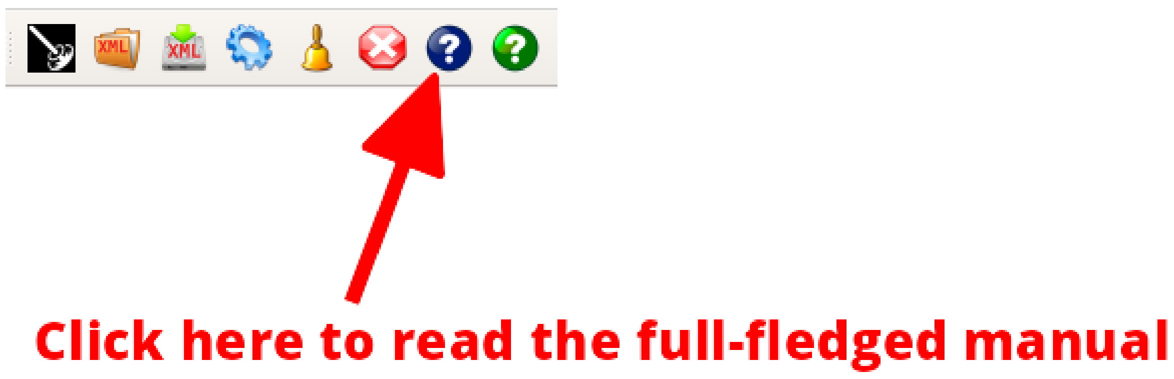
Double left Click here to modify values

1.7.8 Tip 98



Click here to read the iradina-code manual

1.7.9 Tip 99



CHAPTER

TWO

PROGRAMMER'S GUIDE

2.1 Prerequisites

There are some definitions, and links.

1. Iradina[15] code, and its manual.
2. PYTHON[16] 3.5, with packages PyQt5[17], numpy, matplotlib[18], pandas[19], etc. (usually named *py3qt5*).
3. PyInstaller[20] 3.4, free sofware to make iradinaGUI bundle (*only* valid for Linux).
4. 7-zip[21], free sofware to compress/uncompress .7z files (for Windows installations).

Installation needs a PYTHON[22] 3.6 interpreter, which is included in *All-in-one* iradina installations (Linux *and* Windows). see *Python installation Linux* (page 28).

PyInstaller[23] is a program that freezes Python programs in *bundle*, which is **almost** a python package. For more information about *bundle*, see PyInstaller manual[24].

2.2 All-in-one installation

These installations contain in **one** compressed file:

1. the iradinaGUI python scripts.
2. An interpreter PYTHON[25].
3. The Iradina code (GPL), its **source files**, and two executable files, one for Linux and one Windows.
4. The useful Corteo[26] data base (4bits).

Warning: Corteo data base used version in iradinaGUI is **NOT** Version 20160816.

2.3 All-in-one installation Linux

Warning: This is a PyInstaller[27] bundle, installed locally **where users want**. Python interpreter (named py3qt5) is simultaneously installed,

Source tar file *iradinaGUI_bundle_xxxx.tgz* is the **one** compressed archive file of the PyInstaller bundle in *one folder* mode. See more information here[28].

There are two ways to install iradinaGUI:

1. Install iradinaGUI directly, using usual *file manager* functionalities: uncompress tar file in user's choice directory.

[15] https://www.nano.uni-jena.de/en/iradina.html
[16] https://docs.python.org/3.5
[17] https://pypi.org/project/PyQt5
[18] https://matplotlib.org
[19] https://pandas.pydata.org
[20] https://www.pyinstaller.org
[21] https://www.7-zip.org
[22] https://docs.python.org/3.5
[23] https://www.pyinstaller.org
[24] https://pyinstaller.readthedocs.io/en/stable
[25] https://docs.python.org/3.5
[26] http://www.lps.umontreal.ca/~schiette/index.php?n=Recherche.Corteo
[27] https://www.pyinstaller.org
[28] https://pyinstaller.readthedocs.io/en/stable/operating-mode.html#bundling-to-one-folder

2. Install iradinaGUI typing bash command, in usual *terminal*:

```
# install
cd yourChoiceDirectory       # which is really where you want
tar -xf .../iradinaGUI_bundle_xxxx.tgz
# launch
cd iradinaGUI_bundle         # folder name as linux pyinstaller bundle
./iradinaGUI -h              # on line help
./iradinaGUI -g -w ...       # launch GUI
```

2.4 All-in-one installation Windows7-10

Warning:

1. This is **not** a PyInstaller[29] bundle.
2. The **mandatory located** root directory is *C:\Users\Public\iradina*.
3. The **mandatory located** iradinaGUI directory is *C:\Users\Public\iradina\iradinaGUI*.
4. The **mandatory located** Python interpreter py3qt5 directory is *C:\Users\Public\iradina\miniconda3*.
5. The sofware tool to uncompress .7z files is 7-zip[30], which **has to be installed**.

Source .7z file *iradinaGUI_xxxx.7z* is the **one** compressed archive file.

There are two ways to install iradinaGUI:

1. Install iradinaGUI directly, using usual *file manager* functionalities: uncompress .7z file in mandatory *C:\Users\Public* directory.
2. Install iradinaGUI typing DOS command, in usual *(Windows7/10-cmd.exe shell)*:

```
rem install
C:\
cd C:\Users\Public   # this is mandatory location, useful for all users
"C:\Program Files\7-Zip\7z.exe" x .../iradina_xxxx.7z
rem launch
C:\User\Public\iradina\iradinaGUI\LaunchIradinaGUI.bat
```

Note: To launch iradina GUI, you may use Windows shortcut *C:\User\Public\iradina\iradinaGUI\LaunchIradinaGUI(.lnk)*

.

[29] https://www.pyinstaller.org
[30] https://www.7-zip.org

2.5 Development installations

A development installation of iradinaGUI allows programmer improvments. It is a classical usage of Python[31] packages. It needs a directory for Python[32] interpreter (usually named *miniconda3*), and another directory for iradinaGUI scripts (usually named *iradinaGUI*),

In fine, user could find (and use) command *iradinaGUI* directly after a *detar/unzip* installation. Or a *git clone*.

Warning:

1. **Windows7-10** all-in-one installation is a *development installation* of iradinaGUI, which allows programmer improvments.
2. Users find *miniconda3* and *iradinaGUI* directories in parent directory named *C:\Users\Public\iradina*.
3. **Linux** all-in-one installation PyInstaller[33] bundle is **NOT** like that.
4. To get *development installation* **Linux** (freely located) of iradinaGUI, users have to follow the two next chapters.

2.5.1 Python installation Linux

To install python3 (and its mandatory packages PyQt5 etc.) *locally*, we suggest to use miniconda[34]. Note that *miniconda* is windows7-10 compliant.

Note: You may use this Python interpreter for another python scripting code than iradinaGUI.

For information:

1. https://conda.io/miniconda.html
2. https://conda.io/docs/index.html

Example of install *(Linux-bash)*:

```
bash Miniconda3-latest-Linux-x86_64.sh
# -> Miniconda3 will now be installed into this location:
# -> /volatile/common/miniconda3 (for example. It is located as you want.)
# -> Thank you for installing Miniconda3!

export PATH=/volatile/common/miniconda3/bin:$PATH
which conda
# -> /volatile/common/miniconda3/bin/conda

conda create --name py3qt5 python=3 \
   pip jupyter matplotlib numpy pandas pandas-datareader \
   pyqt=5 scipy sympy jsonschema pyyaml libxml2 paramiko
# -> Solving environment: done
# -> Proceed ([y]/n)? y
# -> Downloading and Extracting Packages
# -> To activate this environment, use:
# -> source activate py3qt5

conda info --envs
# -> conda environments:
```

(continues on next page)

[31] https://docs.python.org/3.5
[32] https://docs.python.org/3.5
[33] https://www.pyinstaller.org
[34] http://conda.pydata.org/miniconda.html

(continued from previous page)

```
# ->   base         /volatile/common/miniconda3
# ->   py3qt5       /volatile/common/miniconda3/envs/py3qt5
# ->   etc...

source activate py3qt5
which python
# - > /volatile/common/miniconda3/envs/py3qt5/bin/python
```

2.5.2 iradinaGUI installation Linux

Warning: Python interpreter py3qt5 is supposed to be set and useful in environment path. Usually command *source activate py3qt5* assume that.

Example of install/launch *(Linux-bash)*:

```
cd whereYouWant
tar -xf .../iradinaGUI_xxxx.tgz
cd iradinaGUI
ls -l iradinaGUI        # the launch executable command (is a script python)
which python            # --> py3qt5
./iradinaGUI -h         # on line help
./iradinaGUI -g -w ...  # launch GUI
```

2.5.3 iradinaGUI installation Windows7-10

Warning: Python interpreter py3qt5 is supposed to be set and useful in environment path, at **mandatory** usual location *C:\Users\Public\iradina\miniconda3*. Usually command *conda activate py3qt5* assume that.

Example of install/launch *(Windows7/10-cmd.exe shell)*, using 7-zip[35]:

```
C:\
cd C:\Users\Public\iradina   # this is mandatory location, useful for all users
"C:\Program Files\7-Zip\7z.exe" x .../iradinaGUI_xxxx.7z
cd C:\Users\Public\iradina\iradinaGUI
where python                 # --> py3qt5
python iradinaGUI -h         # on line help
python iradinaGUI -g -w ...  # launch GUI
```

Note: To launch iradina GUI, you may use Windows shortcut *C:\User\Public\iradina\iradinaGUI\LaunchIradinaGUI(.lnk)* .

[35] https://www.7-zip.org

2.6 IradinaGUI configuration

IradinaGUI uses files to store its configuration parameters. It uses ConfigParser[36] package, from The Python Standard Library.

Two configuration files are created or used at iradinaGUI launch, and located at *IRADINAGUI_WORKDIR* directory.

1. file *.../IRADINAGUI_WORKDIR/iradinaGUI_user.cfg*
2. file *.../IRADINAGUI_WORKDIR/iradinaGUI_default.cfg*

Note: User may edit/modify file *iradinaGUI_user.cfg*

2.6.1 Syntax

See https://docs.python.org/3/library/configparser.html

2.6.2 Description

The effective configuration **is a merge** of these two previous files. Parameters in *iradinaGUI_user.cfg* override parameters in *iradinaGUI_default.cfg*.

User will find **all allowed parameters** in systematically up-to-dated *iradinaGUI_default.cfg*.

[36] https://docs.python.org/3/library/configparser.html

2.7 Usage of iradinaGUI

2.7.1 Usage

IradinaGUI usage is a Command Line Interface (CLI[37]), which is Windows *and* Linux compatible.

```
iradinaGUI --[options]
```

Options of iradinaGUI

Useful but *not exhaustive* generic options of *iradinaGUI* CLI.

Option –*help or -h*

Get help as simple text.

```
iradinaGUI --help          # get list of existing options
```

Option –*doc or -d*

Get documentation as browser html.

```
iradinaGUI --doc           # see html doc
```

Option –*verbose or -v*

Change verbosity level (default is 'info').

```
# execute iradinaGUI command in verbose debug mode
iradinaGUI -v debug
```

Option –*workdir or -w*

Change working directory (user data directory). Default is ../IRADINAGUI_WORKDIR

```
# execute iradinaGUI in user choice working directory
iradinaGUI -w .../MY_WORKDIR
```

[37] https://en.wikipedia.org/wiki/Command-line_interface

2.8 Iradina code compilation

IradinaGUI uses a specific version of Iradina code, modified by J.P. Crocombette (cea), which is tagged version 1.1.x for now. This code comes from original version 1.0.8 by Christian Borschel.

Users find two current compiled executable files, which are used by iradinaGUI, located at *.../iradinaGUI/iradinaCode*.

1. *iradina_mingw64.exe*, compiled by MinGW[38] gcc compiler, for Windows (64 bits).
2. *iradina_linux64.exe*, compiled by GNU[39] gcc[40] compiler, for Linux (64 bits).

A development installation of Iradina code allows programmer improvments. The following chapters explain Iradina code compilation processes.

2.8.1 Iradina code sources

With GPL licence, sources are available in iradinaGUI directory tree, located at *.../iradinaGUI/iradinaCodes/iradina_cea*.

User find also the useful Corteo[41] data base, located at *.../iradinaGUI/iradinaCodes/data_4bit*

```
.../iradinaCodes > tree
.
├── data_4bit
│   ├── 10.asp
│   ├── 11.asp
etc.
│   ├── corteo.mat
│   └── erfinv.dat
├── doc
│   ├── 20140804_iradina_manual.pdf
│   ├── Corteo20160816.pdf
│   ├── iradina-1-s2.0-S0168583X11006318-main.pdf
│   └── Iradina_tuto_installation.pdf
├── iradina_cea
│   ├── compileIradina.bat
│   ├── fileio.c
│   ├── fileio.h
│   ├── fromcorteo.c
│   ├── fromcorteo.h
│   ├── geometry.c
│   ├── geometry.h
│   ├── indexvalues6bit.h
│   ├── indexvalues.h
│   ├── iradina.c
│   ├── iradina.h
│   ├── license.txt
│   ├── makefile_cea
│   ├── target.c
│   ├── target.h
│   ├── transport.c
│   ├── transport.h
│   ├── utils.c
│   └── utils.h
├── compileIradina.lnk
└── README.txt
```

[38] http://www.mingw.org
[39] https://www.gnu.org/home.en.html
[40] https://www.gnu.org/software/gcc
[41] http://www.lps.umontreal.ca/~schiette/index.php?n=Recherche.Corteo

2.8.2 Iradina code compilation Linux

Example of compilation *(Linux-bash)*:

```
# this is your location
cd .../iradinaGUI/iradinaCodes/iradina_cea
# verifications
cat README.txt
# compilation
make -f makefile_cea clean
make -f makefile_cea iradina
make -f makefile_cea installGUI  # install executable in iradinaGUI/iradinaCode
```

2.8.3 Iradina code compilation Windows7-10

Warning:

1. MinGW[42] is supposed to be set and useful in environment path, at an usual location *C:\MinGW* (for example).
2. Git-windows[43] is supposed to be set and useful in environment path, at an usual location *C:\Program Files\Git* (for example). In order to use like-Linux commands.

Example of compilation *(Windows7/10-cmd.exe shell)*:

```
# this is mandatory location
C:\
cd C:\Users\Public\iradina\iradinaGUI\iradinaCodes\iradina_cea
# verifications
where make                        # --> C:\MinGW\bin\make.exe
where gcc                         # --> C:\MinGW\bin\gcc.exe
where uname                       # --> C:\Program Files\Git\usr\bin\uname.exe
# compilation
make -f makefile_cea clean
make -f makefile_cea iradina
make -f makefile_cea installGUI  # install executable in iradinaGUI/iradinaCode
```

Note: To launch Iradina code compilation, you may use Windows shortcut *C:\User\Public\iradina\iradinaGUI\iradinaCodes\compileIradina(.lnk)* .

[42] http://www.mingw.org
[43] https://git-scm.com

2.9 Documentation

2.9.1 Doc consultation

To display iradinaGUI html documentation in your web browser *firefox*, or *else*. The initial entry file is located at *iradinaGUI/doc/build/html/index.html*.

```
# Linux bash, as an example
cd .../iradinaGUI
firefox doc/build/html/index.html &
# or as CLI_
iradinaGUI --doc
```

2.9.2 Doc modification

To modify iradinaGUI documentation with simple editor *pluma*, or else.

Read the manual, see http://www.sphinx-doc.org/en/stable/tutorial.html, or may be copy/paste from 'Show Source' item.

```
# Linux bash, as an example
cd ...iradinaGUI/
tmp=$(find doc -name "*.rst")
pluma $tmp &
```

2.9.3 Doc compilation Linux

On a Linux system, to compile iradinaGUI html documentation, programmers use installed GNU[44] *make*, and SPHINX[45].

Warning: To make documentation **pdf** programmers needs installed *texlive* package (preferably up to date version). See: https://www.tug.org/texlive/quickinstall.html

```
cd ...iradinaGUI/doc
cat README  # read some environment setup information
#  ... and read it
make
  Please use `make <target>' where <target> is one of
  html       to make standalone HTML files
  dirhtml    to make HTML files named index.html in directories
  singlehtml to make a single large HTML file
  pickle     to make pickle files
  json       to make JSON files
  htmlhelp   to make HTML files and a HTML help project
  qthelp     to make HTML files and a qthelp project
  devhelp    to make HTML files and a Devhelp project
  epub       to make an epub
  latex      to make LaTeX files, you can set PAPER=a4 or PAPER=letter
  latexpdf   to make LaTeX files and run them through pdflatex
  text       to make text files
  man        to make manual pages
  changes    to make an overview of all changed/added/deprecated items
  linkcheck  to check all external links for integrity
```

(continues on next page)

[44] https://www.gnu.org/home.en.html
[45] http://sphinx-doc.org

(continued from previous page)

```
  doctest    to run all doctests embedded in the documentation (if enabled)

# and then
make html      # make html doc
make latexpdf  # make pdf doc
```

CHAPTER

THREE

FREQUENTLY ASKED QUESTIONS

3.1 Add an Item in this FAQ

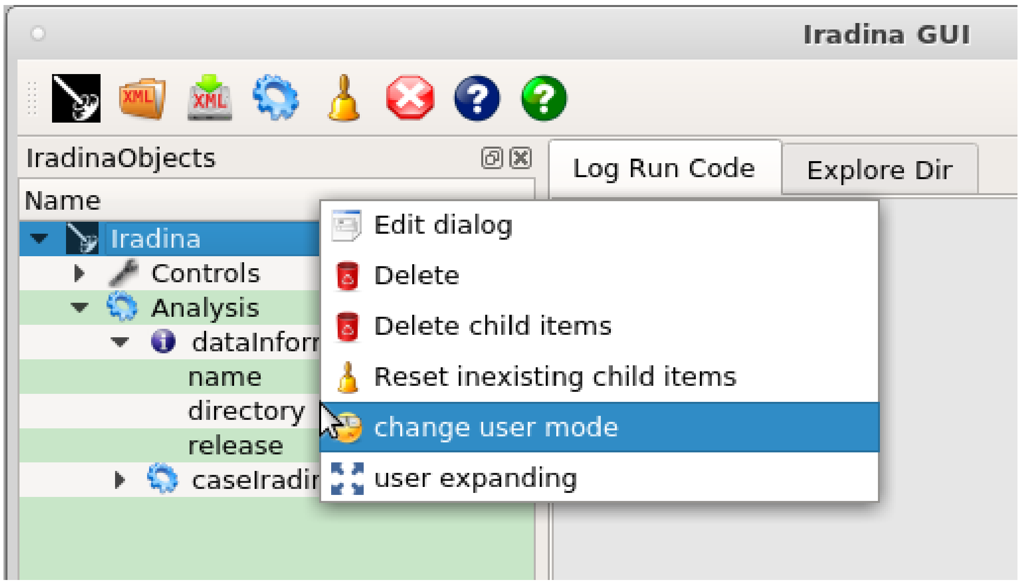
Edit *iradinaGUI/doc/src/FAQ.rst* and read documentation *Doc compilation Linux* (page 34).

3.2 FAQ

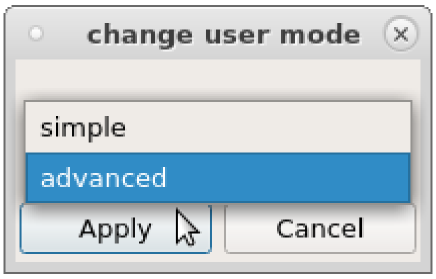
3.2.1 Why there is NOT all options of Iradina code in IradinaObjects tree ?

Default user mode is set to **simple**, for beginners use.

Select action *change user mode* in contextual menu of *Iradina* node of tree (which is first line/node of tree).



Then Apply as **advanced**.



User can edit/modify value *General.usermode* parameter in the *IradinaGUI configuration* (page 30), to get a permanent setting.

3.2.2 Next question ?

Here there is your next answer etc.

CHAPTER

FOUR

CODE DOCUMENTATION

4.1 Code documentation

4.1.1 iradinapy

iradinapy package

Subpackages

iradinapy.colorama package

Submodules

iradinapy.colorama.ansi module

This module generates ANSI character codes to printing colors to terminals. See: http://en.wikipedia.org/wiki/ANSI_escape_code

class `iradinapy.colorama.ansi.`**`AnsiBack`**

Bases: *`iradinapy.colorama.ansi.AnsiCodes`* (page 40)

`BLACK = 40`

`BLUE = 44`

`CYAN = 46`

`GREEN = 42`

`LIGHTBLACK_EX = 100`

`LIGHTBLUE_EX = 104`

`LIGHTCYAN_EX = 106`

`LIGHTGREEN_EX = 102`

`LIGHTMAGENTA_EX = 105`

`LIGHTRED_EX = 101`

`LIGHTWHITE_EX = 107`

`LIGHTYELLOW_EX = 103`

`MAGENTA = 45`

`RED = 41`

`RESET = 49`

`WHITE = 47`

`YELLOW = 43`

class `iradinapy.colorama.ansi.`**`AnsiCodes`**

Bases: `object`

class `iradinapy.colorama.ansi.`**`AnsiCursor`**

Bases: `object`

`BACK`(*n=1*)

`DOWN`(*n=1*)

`FORWARD`(*n=1*)

`POS`(*x=1*, *y=1*)

UP(*n=1*)

class iradinapy.colorama.ansi.**AnsiFore**

Bases: *iradinapy.colorama.ansi.AnsiCodes* (page 40)

BLACK = 30

BLUE = 34

CYAN = 36

GREEN = 32

LIGHTBLACK_EX = 90

LIGHTBLUE_EX = 94

LIGHTCYAN_EX = 96

LIGHTGREEN_EX = 92

LIGHTMAGENTA_EX = 95

LIGHTRED_EX = 91

LIGHTWHITE_EX = 97

LIGHTYELLOW_EX = 93

MAGENTA = 35

RED = 31

RESET = 39

WHITE = 37

YELLOW = 33

class iradinapy.colorama.ansi.**AnsiStyle**

Bases: *iradinapy.colorama.ansi.AnsiCodes* (page 40)

BRIGHT = 1

DIM = 2

NORMAL = 22

RESET_ALL = 0

iradinapy.colorama.ansi.**clear_line**(*mode=2*)

iradinapy.colorama.ansi.**clear_screen**(*mode=2*)

iradinapy.colorama.ansi.**code_to_chars**(*code*)

iradinapy.colorama.ansi.**set_title**(*title*)

iradinapy.colorama.ansitowin32 module

class iradinapy.colorama.ansitowin32.**AnsiToWin32**(*wrapped*, *convert=None*, *strip=None*, *autoreset=False*)

Bases: object

Implements a 'write()' method which, on Windows, will strip ANSI character sequences from the text, and if outputting to a tty, will convert them into win32 function calls.

ANSI_CSI_RE = re.compile('\x01?\x1b\\[((?:\\d|;)*)([a-zA-Z])\x02?')

ANSI_OSC_RE = re.compile('\x01?\x1b\\]((?:.|;)*?)(\x07)\x02?')

call_win32(*command*, *params*)

convert_ansi(*paramstring*, *command*)

convert_osc(*text*)

extract_params(*command*, *paramstring*)

get_win32_calls()

reset_all()

should_wrap()
True if this class is actually needed. If false, then the output stream will not be affected, nor will win32 calls be issued, so wrapping stdout is not actually required. This will generally be False on non-Windows platforms, unless optional functionality like autoreset has been requested using kwargs to init()

write(*text*)

write_and_convert(*text*)
Write the given text to our wrapped stream, stripping any ANSI sequences from the text, and optionally converting them into win32 calls.

write_plain_text(*text*, *start*, *end*)

class iradinapy.colorama.ansitowin32.**StreamWrapper**(*wrapped*, *converter*)
Bases: object

Wraps a stream (such as stdout), acting as a transparent proxy for all attribute access apart from method 'write()', which is delegated to our Converter instance.

write(*text*)

iradinapy.colorama.ansitowin32.**is_a_tty**(*stream*)

iradinapy.colorama.ansitowin32.**is_stream_closed**(*stream*)

iradinapy.colorama.initialise module

iradinapy.colorama.initialise.**colorama_text**(**args*, ***kwargs*)

iradinapy.colorama.initialise.**deinit**()

iradinapy.colorama.initialise.**init**(*autoreset=False*, *convert=None*, *strip=None*, *wrap=True*)

iradinapy.colorama.initialise.**reinit**()

iradinapy.colorama.initialise.**reset_all**()

iradinapy.colorama.initialise.**wrap_stream**(*stream*, *convert*, *strip*, *autoreset*, *wrap*)

iradinapy.colorama.win32 module

iradinapy.colorama.win32.**SetConsoleTextAttribute**(**_*)

iradinapy.colorama.win32.**winapi_test**(**_*)

iradinapy.colorama.winterm module

class iradinapy.colorama.winterm.**WinColor**

Bases: `object`

BLACK = 0

BLUE = 1

CYAN = 3

GREEN = 2

GREY = 7

MAGENTA = 5

RED = 4

YELLOW = 6

class iradinapy.colorama.winterm.**WinStyle**

Bases: `object`

BRIGHT = 8

BRIGHT_BACKGROUND = 128

NORMAL = 0

class iradinapy.colorama.winterm.**WinTerm**

Bases: `object`

back (*back=None*, *light=False*, *on_stderr=False*)

cursor_adjust (*x*, *y*, *on_stderr=False*)

erase_line (*mode=0*, *on_stderr=False*)

erase_screen (*mode=0*, *on_stderr=False*)

fore (*fore=None*, *light=False*, *on_stderr=False*)

get_attrs ()

get_position (*handle*)

reset_all (*on_stderr=None*)

set_attrs (*value*)

set_console (*attrs=None*, *on_stderr=False*)

set_cursor_position (*position=None*, *on_stderr=False*)

set_title (*title*)

style (*style=None*, *on_stderr=False*)

Module contents

iradinapy.example package

Submodules

iradinapy.example.essai_logging_1 module

http://sametmax.com/ecrire-des-logs-en-python/ https://docs.python.org/3/library/time.html#time.strftime essai utilisation logger plusieurs handler format different

> /usr/lib/python2.7/logging/__init__.pyc
>
> init MyLogger, fmt='%(asctime)s :: %(levelname)-8s :: %(message)s', level='20'
>
> 2018-03-11 18:51:21 :: INFO :: test logger info 2018-03-11 18:51:21 :: WARNING :: test logger warning 2018-03-11 18:51:21 :: ERROR :: test logger error 2018-03-11 18:51:21 :: CRITICAL :: test logger critical
>
> init MyLogger, fmt='None', level='10'
>
> 2018-03-11 18:51:21 :: DEBUG :: test logger debug test logger debug 2018-03-11 18:51:21 :: INFO :: test logger info test logger info 2018-03-11 18:51:21 :: WARNING :: test logger warning test logger warning 2018-03-11 18:51:21 :: ERROR :: test logger error test logger error 2018-03-11 18:51:21 :: CRITICAL :: test logger critical test logger critical

`iradinapy.example.essai_logging_1.`**`getMyLogger`**`()`

`iradinapy.example.essai_logging_1.`**`initMyLogger`**`(`*fmt=None*, *level=None*`)`

`iradinapy.example.essai_logging_1.`**`testLogger1`**`()`

iradinapy.example.essai_logging_2 module

http://sametmax.com/ecrire-des-logs-en-python/ https://docs.python.org/3/library/time.html#time.strftime

essai utilisation logger un handler format different sur info() pas de format et su other format

> /usr/lib/python2.7/logging/__init__.pyc
>
> init MyLogger, fmt='%(asctime)s :: %(levelname)-8s :: %(message)s', level='20'
>
> test logger info 2018-03-11 18:51:51 :: WARNING :: test logger warning 2018-03-11 18:51:51 :: ERROR :: test logger error 2018-03-11 18:51:51 :: CRITICAL :: test logger critical

`class` `iradinapy.example.essai_logging_2.`**`MyFormatter`**`(`*fmt=None*, *datefmt=None*, *style='%'*`)`

> Bases: `logging.Formatter`
>
> **`format`**`(`*record*`)`
>
> > Format the specified record as text.
> >
> > The record's attribute dictionary is used as the operand to a string formatting operation which yields the returned string. Before formatting the dictionary, a couple of preparatory steps are carried out. The message attribute of the record is computed using LogRecord.getMessage(). If the formatting string uses the time (as determined by a call to usesTime(), formatTime() is called to format the event time. If there is exception information, it is formatted using formatException() and appended to the message.

`iradinapy.example.essai_logging_2.`**`getMyLogger`**`()`

`iradinapy.example.essai_logging_2.`**`initMyLogger`**`(`*fmt=None*, *level=None*`)`

`iradinapy.example.essai_logging_2.`**`testLogger1`**`()`

Module contents

Submodules

iradinapy.abcdExpression module

from ‘3(a2bc)’ to ‘abbcabbcabbc’ without regexp, not recursive for not smart poor people

`iradinapy.abcdExpression.`**`getIndiceFromChar`**(*aChar*)
    returns 0 for ‘a’, 1 for ‘b’ etc., max is ‘z’

`iradinapy.abcdExpression.`**`toAbcd`**(*aStr*, *verbose=False*, *details=False*)
    ‘10(abc)’ to return ‘10*(a+b+c)’ raise exception if problem

`iradinapy.abcdExpression.`**`toEval0123`**(*aStr*, *verbose=False*)
    ‘3(a2bc)’ to return [0,1,1,2,0,1,1,2,0,1,1,2] for ‘abbcabbcabbc’ raise exception if problem

`iradinapy.abcdExpression.`**`toEvalAbcd`**(*aStr*, *verbose=False*)
    ‘3(a2bc)’ to return ‘abbcabbcabbc’ raise exception if problem

`iradinapy.abcdExpression.`**`toEvalAbcdForTooltip`**(*aStr*, *length=20*)
    set results in lenght characters lines

iradinapy.analysisIra module

`class` `iradinapy.analysisIra.`**`AnalysisIra`**
    Bases: `xyzpy.baseXyz._XyzConstrainBase`

    `appendHistoryFileManager`(*action*)

    `createDocLaunch`()

    `getActionsContextMenu`()

    `gitCommit`()

    `gitkLaunch`()

    `isHidden`(*nameAttr*)
        to know if attribute is currently displayed in treeView and other dialog widget

    `packageLaunch`()

    `postTreatments`()

    `printROOTContext`()

    `runPythonCode`()

    `searchURANIEMethod`()

    `setDefaultValues`()
        not virtual, could be used

    `toFileIra`()

    `updateRootlogonLaunch`()

`class` `iradinapy.analysisIra.`**`AttributeDataFrameIra`**(*value=None*)
    Bases: `xyzpy.intFloatListXyz.StrXyz`

    `drawGraphic`()

    `drawUnivariate`()

    `getActionsContextMenu`()

    `getAttributeName`()

getExpressionsInParent()

getFileInParent()

class iradinapy.analysisIra.**DataInformationsIra**

Bases: xyzpy.baseXyz._XyzConstrainBase

general informations about

getEtudeWorkdir()

return as os.path.join(directory,name)

getEtudeWorkdirBrut()

return as ${IRADINAGUI_WORKDIR}

getVersion()

isHidden(*nameAttr*)

to know if attribute is currently displayed in treeView and other dialog widget

setDefaultValues()

not virtual, could be used

class iradinapy.analysisIra.**DataIra**

Bases: xyzpy.baseXyz._XyzConstrainBase

DataFrame

copyFileInData()

drawDialogDataServer()

getActionsContextMenu()

getAllAttributesName()

getNameExpanded()

setAttributes()

append attributes from '#COLUMN_NAMES' from file .dat, ordered as useful in uranie

class iradinapy.analysisIra.**DataManagerIra**(*nameObject=''*)

Bases: *iradinapy.analysisIra.ListOfFileViewerXyz* (page 47)

copyAllFileInData()

getActionsContextMenu()

class iradinapy.analysisIra.**ExpressionIra**(*value=None*)

Bases: xyzpy.intFloatListXyz.ExpressionXyz

createEditorData(*parent=None*)

drawGraphic()

drawUnivariate()

getActionsContextMenu()

no browse

getAllAttributesNameInParents()

getAttributeName()

synonym for coherency with AttributeIra

getExpressionsInParent()

getFileInParent()

getName()

isNameUnique(*name*)

test unicity of name in DataIra search in lists DataIra.attributes and DataIra.expressions

isValidExpression(*value*)
test if expression is valid in a uranie TDataServer

class iradinapy.analysisIra.**FileIra**(*value=None*)
Bases: `xyzpy.intFloatListXyz.FileViewerXyz`

initialize type of files extension for interest .C etc.

class iradinapy.analysisIra.**FunctionIra**(*value=None*)
Bases: *iradinapy.analysisIra.FileIra* (page 47)

initialize type of files extension for function files

class iradinapy.analysisIra.**HistoryFileManagerXyz**
Bases: `xyzpy.baseXyz._XyzConstrainBase`

store in string for xml save history on some files, with hashing control as cp, mv etc... actions from gui/model actions one line by action

appendHistoryAction(*action*)

appendHistoryCopyOf(*originFile*, *newFile*)

clearHistory()

getCompleteFileName(*aFile*)

getFileHash(*aFile*)

getIdentFile(*aFile*)

isHidden(*nameAttr*)
to know if attribute is currently displayed in treeView and other dialog widget

setDefaultValues()
not virtual, could be used

class iradinapy.analysisIra.**LibraryIra**(*value=None*)
Bases: *iradinapy.analysisIra.FileIra* (page 47)

initialize type of files extension for libraries files .C .so

class iradinapy.analysisIra.**ListOfAttributeIra**(*nameObject=''*)
Bases: `xyzpy.baseXyz.ListOfBaseXyz`

getAllAttributesName()

class iradinapy.analysisIra.**ListOfExpressionIra**(*nameObject=''*)
Bases: `xyzpy.baseXyz.ListOfBaseXyz`

getAllAttributesName()

getExpressions()

class iradinapy.analysisIra.**ListOfFileViewerXyz**(*nameObject=''*)
Bases: `xyzpy.baseXyz.ListOfBaseXyz`

base class, used only inheritage, modify _allowedClasses etc

addItem(*aClass=None*)
override method ListOfBaseXyz

addItems(*aClass=None*)

addItemsSlot(*status*, *aClass=None*)
new pyqt5 state override in lambda

browseViewerDialog()

browseViewerExecOnApply(*selectedFiles*)
part of Apply on browseViewerDialog

getActionsContextMenu()

getDirectory()
could be dynamic in inheritage

getNamesExpanded()

getNoLocal()

getTargetDirectory()
could be dynamic in inheritage

class iradinapy.analysisIra.**ListOfFunctionIra**(*nameObject=''*)
Bases: *iradinapy.analysisIra.ListOfFileViewerXyz* (page 47)

class iradinapy.analysisIra.**ListOfLibraryIra**(*nameObject=''*)
Bases: *iradinapy.analysisIra.ListOfFileViewerXyz* (page 47)

class iradinapy.analysisIra.**ListOfMacroIra**(*nameObject=''*)
Bases: *iradinapy.analysisIra.ListOfFileViewerXyz* (page 47)

class iradinapy.analysisIra.**ListOfUserFileIra**(*nameObject=''*)
Bases: *iradinapy.analysisIra.ListOfFileViewerXyz* (page 47)

getDirectory()
dynamic override _directory

getTargetDirectory()
dynamic override _targetDirectory

class iradinapy.analysisIra.**MacroIra**(*value=None*)
Bases: *iradinapy.analysisIra.FileIra* (page 47)

initialize type of files extension for macro .C .py

class iradinapy.analysisIra.**MacroManagerIra**
Bases: xyzpy.baseXyz._XyzConstrainBase

getNamesExpanded()

getNoLocal()

setDefaultValues()
not virtual, could be used

class iradinapy.analysisIra.**UserFileIra**(*value=None*)
Bases: xyzpy.intFloatListXyz.FileViewerXyz

initialize type of files extension for user interest

iradinapy.analysisIra.**drawGraphic**(*self*)
common method draw histogram for classes as AttributeIra or ExpressionIra

iradinapy.analysisIra.**drawUnivariate**(*self*)
common method draw univariate for classes as AttributeIra or ExpressionIra

iradinapy.analysisIra.**fn_heterogenous_random_multiple_materials**(*case*, *aStream*)

iradinapy.analysisIra.**fn_homogenous_one_material**(*case*, *aStream*, *options={}*)

iradinapy.analysisIra.**fn_multiLayer_multiple_materials**(*case*, *aStream*, *options={}*)

iradinapy.analysisIra.**getCurrentRowColumn**(*indiceCurrent*, *nbmaxGrid*, *rowBefore=True*)
returns current (row, column) in nbmax elements in grid for indice current (0 to nbmaxGrid)

iradinapy.analysisIra.**getDefaultRowColumn**(*nb*)
return (row, column)

`iradinapy.analysisIra.`**`getRandomConcMaterial`**(*irandoms*)
returns random indice in Materials (using TargetConcentration) irandoms is list as cumul of TargetConcentration

`iradinapy.analysisIra.`**`get_value_random_multiple_materials`**(*case*)
return random value for integer indice in Materials using getRandomConcMaterial

`iradinapy.analysisIra.`**`join`**(**v*)
as os.path.join but set antislash as slash, even for windows, keep windows 'c:'

`iradinapy.analysisIra.`**`normalize`**(*aList*)

`iradinapy.analysisIra.`**`toFileCompositionIn`**(*case, aStream, name*)
see 'Creating new composition file' line 1222 in utils.C

`iradinapy.analysisIra.`**`toFileConfigurationIn`**(*case, aStream, name=''*)

`iradinapy.analysisIra.`**`toFileMaterialIn`**(*case, aStream, name=''*)

`iradinapy.analysisIra.`**`toFileStructureIn`**(*case, aStream, name=''*)

`iradinapy.analysisIra.`**`toValue`**(*aStr, name, value*)

iradinapy.caseIradina module

class `iradinapy.caseIradina.`**`BeamSpreadIra`**(*value=None*)
Bases: `xyzpy.intFloatListXyz.FloatPosXyz`

float positive with default 1.

class `iradinapy.caseIradina.`**`BoolFalseIra`**(*value=None*)
Bases: `xyzpy.intFloatListXyz.BoolXyz`

['False', 'True'] with write config file iradina strCfg [0, 1] default False

`strCfg`()

class `iradinapy.caseIradina.`**`BoolTrueIra`**(*value=None*)
Bases: *`iradinapy.caseIradina.BoolFalseIra`* (page 49)

['False', 'True'] with write config file iradina strCfg [0, 1] default True

class `iradinapy.caseIradina.`**`CaseIra`**
Bases: `xyzpy.baseXyz._XyzConstrainBase`

general informations about case and for launch iradina

`isHidden`(*nameAttr*)
to know if attribute is currently displayed in treeView and other dialog widget

`setDefaultValues`()
not virtual, could be used

class `iradinapy.caseIradina.`**`CellCountxIra`**(*value=None*)
Bases: `xyzpy.intFloatListXyz.IntSupEq1Xyz`

int positive with default 100

class `iradinapy.caseIradina.`**`CellCountyIra`**(*value=None*)
Bases: `xyzpy.intFloatListXyz.IntSupEq1Xyz`

int positive with default 1

class `iradinapy.caseIradina.`**`CellCountzIra`**(*value=None*)
Bases: `xyzpy.intFloatListXyz.IntSupEq1Xyz`

int positive with default 1

class `iradinapy.caseIradina.`**`CellDepthxIra`**(*value=None*)
Bases: `xyzpy.intFloatListXyz.FloatPosXyz`

float positive with default 1000. (nm)

`toXml`(***kwargs*)
set tooltip_1 attibute xml for tooltip as long name element (value column 1)

class `iradinapy.caseIradina.`**`CellMultiLayerxIra`**(*value=None*)
Bases: `xyzpy.intFloatListXyz.StrXyz`

'abcd' values for multiLayer description

`getCount`()
returns len(self) as 8 for '2(abcd)'

`getIndiceFromChar`(*aChar*)
returns 0 for 'a', 1 for 'b' etc.

`getMultiLayerMaterial`(*indice*, *verbose=False*)

`toEval0123`()
returns [0,1,2,3,0,1,2,3] for '2(abcd)'

`toEvalAbcd`()
returns 'abcdabcd' for '2(abcd)'

class `iradinapy.caseIradina.`**`CellSizexIra`**(*value=None*)
Bases: `xyzpy.intFloatListXyz.FloatPosXyz`

float positive with default 10.

class `iradinapy.caseIradina.`**`CellSizeyIra`**(*value=None*)
Bases: `xyzpy.intFloatListXyz.FloatPosXyz`

float positive with default 100.

class `iradinapy.caseIradina.`**`CellSizezIra`**(*value=None*)
Bases: `xyzpy.intFloatListXyz.FloatPosXyz`

float positive with default 100.

class `iradinapy.caseIradina.`**`CompositionFileTypeIra`**(*value=None*)
Bases: `xyzpy.intFloatListXyz.StrInListXyz`

If the file is just one column of values then FileType should be set to 1. If the file contains 4 columns (x, y, z, value) then set it to 0.

`strCfg`()
get index from '(i) blah blah'

class `iradinapy.caseIradina.`**`ConcentrationIra`**(*value=None*, *minMax=None*)
Bases: `xyzpy.intFloatListXyz.FloatRangeXyz`

initial value as 1.

class `iradinapy.caseIradina.`**`DensityIra`**(*value=None*)
Bases: `xyzpy.intFloatListXyz.FloatPosXyz`

float positive with default 0. supposed g/cm3, have to convert to at/cm3

`getActionsContextMenu`()
append action 'Set defaut value'

`getCalculatedValue`()
prorata ElementConc(s) and Components densities a trivial approximation

`normalize`(*aList*)

`setCalculatedValue`()

strCfg()
iradina needs density of the material in atoms/cm3

toAtomCm3()
prorata ElementConc(s) and Components atomic weights

toXml(***kwargs*)
set tooltip_1 attibute xml for tooltip as long name element (value column 1)

class iradinapy.caseIradina.**DensityTargetComponentIra**(*value=None*)
Bases: xyzpy.intFloatListXyz.FloatPosXyz

TODO could set default density from wiki localMendeleiev or user config

getActionsContextMenu()
append action 'Set defaut value'

getUserConfigValue()

getWikipediaValue()

setDefaultDensityUser()

setDefaultDensityWiki()

class iradinapy.caseIradina.**DisplayIntervalIra**(*value=None*)
Bases: xyzpy.intFloatListXyz.IntSupEq1Xyz

int positive with default 20

class iradinapy.caseIradina.**ElementCountIra**(*value=None*)
Bases: xyzpy.intFloatListXyz.IntRangeXyz

as define MAX_NO_MATERIALS 20 as Maximum number of different materials

class iradinapy.caseIradina.**ElementReplEnergyIra**(*value=None*)
Bases: xyzpy.intFloatListXyz.FloatXyz

float positive with default value -1 as None

class iradinapy.caseIradina.**FlightLengthTypeIra**(*value=None*)
Bases: xyzpy.intFloatListXyz.StrInListXyz

strCfg()
get index from '(i) blah blah'

class iradinapy.caseIradina.**FloatListIra**(*value=None*)
Bases: xyzpy.intFloatListXyz.StrXyz

accept [val1, val2, val3]

class iradinapy.caseIradina.**IonAngleIra**(*value=None*)
Bases: xyzpy.intFloatListXyz.IntRangeXyz

angle incidence xy (degree) integer range -90 to 90 default 0.

toVx()

toVy()

toXml(***kwargs*)
kwarg are for optional future option of added details in xml tree

class iradinapy.caseIradina.**IonBeamIra**
Bases: xyzpy.baseXyz._XyzConstrainBase

general informations about beam iradina

isHidden(*nameAttr*)
to know if attribute is currently displayed in treeView and other dialog widget

class iradinapy.caseIradina.**IonDistributionIra**(*value=None*)
Bases: xyzpy.intFloatListXyz.StrInListXyz

strCfg()
get index from '(i) blah blah'

class iradinapy.caseIradina.**IonDoseIra**(*value=None*)
Bases: xyzpy.intFloatListXyz.FloatXyz

float positive with default value -1 as None

class iradinapy.caseIradina.**IonE0Ira**(*value=None*)
Bases: xyzpy.intFloatListXyz.FloatPosXyz

float positive with default 10e3 keV

class iradinapy.caseIradina.**IonMIra**(*value=None*)
Bases: xyzpy.intFloatListXyz.FloatPosXyz

Ion mass (g/mol) float positive with default 1.

class iradinapy.caseIradina.**IonVxIra**(*value=None*)
Bases: xyzpy.intFloatListXyz.FloatPosXyz

vector incidence x float positive with default 1.

class iradinapy.caseIradina.**IonVyIra**(*value=None*)
Bases: xyzpy.intFloatListXyz.FloatPosXyz

vector incidence y float positive with default 0.

class iradinapy.caseIradina.**IonVzIra**(*value=None*)
Bases: xyzpy.intFloatListXyz.FloatPosXyz

vector incidence z float positive with default 0

class iradinapy.caseIradina.**IsotopeIra**
Bases: xyzpy.baseXyz._XyzConstrainBase

general informations about IsotopeIra

browseElement()

checkValues(*verbose=True*)

getActionsContextMenu()
append action 'Append file projectile'

isHidden(*nameAttr*)
to know if attribute is currently displayed in treeView and other dialog widget

on_attributesChange(*verbose=False*)

setDefaultValues()
not virtual, could be used

class iradinapy.caseIradina.**ListOfMaterialIra**(*nameObject=''*)
Bases: xyzpy.baseXyz.ListOfBaseXyz

Important note: it is strongly recommended NOT to create one file with all materials that you know for all of your simulations! You should just include the materials you really need for the current simulation, because iradina will create 2.6 MByte scattering matrices for every possible combination of two elements in the target! So the memory usage increases with the square of the number of different elements! If your materials contain 92 different elements, iradina needs 22 GByte of memory in the 4-MSB version or 352 GByte in the 6-MSB version

getAllAttributesName()

class iradinapy.caseIradina.**ListOfTargetComponentsIra**(*nameObject=''*)
Bases: xyzpy.baseXyz.ListOfBaseXyz

class iradinapy.caseIradina.**MaterialIra**
Bases: xyzpy.baseXyz._XyzConstrainBase

general informations about material of target iradina

getElementCount()

getElementsConc()

getElementsDispEnergy()

getElementsLattEnergy()

getElementsM()

getElementsReplEnergy()

getElementsSurfEnergy()

getElementsSymbol()

getElementsZ()

isHidden(*nameAttr*)
to know if attribute is currently displayed in treeView and other dialog widget

normalize(*aList*)

class iradinapy.caseIradina.**MaterialNameIra**(*value=None*)
Bases: xyzpy.intFloatListXyz.StrXyz

string no more 24 characters target.c: if(strlen(MaterialName)>=25){MaterialName[24]='';}

class iradinapy.caseIradina.**MaxNoIonIra**(*value=None*)
Bases: xyzpy.intFloatListXyz.IntPosXyz

int positive with default 20000

class iradinapy.caseIradina.**MinEnergyIra**(*value=None*)
Bases: xyzpy.intFloatListXyz.FloatPosXyz

float positive with default 5.

class iradinapy.caseIradina.**NormalizeOutputIra**(*value=None*)
Bases: xyzpy.intFloatListXyz.StrInListXyz

strCfg()
get index from '(i) blah blah'

class iradinapy.caseIradina.**Seed1Ira**(*value=None*)
Bases: xyzpy.intFloatListXyz.IntPosXyz

int positive with default 123

class iradinapy.caseIradina.**Seed2Ira**(*value=None*)
Bases: xyzpy.intFloatListXyz.IntPosXyz

int positive with default 456

class iradinapy.caseIradina.**SeedIra**(*value=None*)
Bases: xyzpy.intFloatListXyz.IntPosXyz

python random.seed(a=None) to initialize internal state of the random number generator as None or no argument seeds from current time or from an operating system specific randomness source if available

class iradinapy.caseIradina.**SimulationIra**
Bases: xyzpy.baseXyz._XyzConstrainBase

general informations about simulation iradina

isHidden(*nameAttr*)
to know if attribute is currently displayed in treeView and other dialog widget

class iradinapy.caseIradina.**SimulationTypeIra**(*value=None*)
Bases: xyzpy.intFloatListXyz.StrInListXyz

strCfg()
get index from '(i) blah blah'

class iradinapy.caseIradina.**StatusUpdateIntervalIra**(*value=None*)
Bases: xyzpy.intFloatListXyz.IntPosXyz

int positive with default 10000

class iradinapy.caseIradina.**StorageIntervalIra**(*value=None*)
Bases: xyzpy.intFloatListXyz.IntPosXyz

int positive with default 2000

class iradinapy.caseIradina.**StorePathLimitIra**(*value=None*)
Bases: xyzpy.intFloatListXyz.IntSupEq1Xyz

int positive with default 50

class iradinapy.caseIradina.**StorePathLimitRecoilsIra**(*value=None*)
Bases: xyzpy.intFloatListXyz.IntXyz

int positive with default -1 as None

class iradinapy.caseIradina.**StragglingModelIra**(*value=None*)
Bases: xyzpy.intFloatListXyz.StrInListXyz

strCfg()
get index from '(i) blah blah'

class iradinapy.caseIradina.**StructureIra**
Bases: xyzpy.baseXyz._XyzConstrainBase

general informations about target iradina

getCellSizeX()

getMultiLayerMaterial(*indice*)

get_cell_count_x()

isHidden(*nameAttr*)
to know if attribute is currently displayed in treeView and other dialog widget

class iradinapy.caseIradina.**TargetComponentIra**
Bases: xyzpy.baseXyz._XyzConstrainBase

general informations about TargetComponentIra

browseElement()

checkValues(*verbose=True*)

getActionsContextMenu()
append action 'Append file projectile'

isHidden(*nameAttr*)
to know if attribute is currently displayed in treeView and other dialog widget

on_attributesChange(*verbose=False*)

setDefaultValues()
not virtual, could be used

class iradinapy.caseIradina.**TargetIra**
Bases: xyzpy.baseXyz._XyzConstrainBase

general informations about target iradina

isHidden(*nameAttr*)
  to know if attribute is currently displayed in treeView and other dialog widget

class iradinapy.caseIradina.**TypeCompositionIra**(*value=None*)
  Bases: `xyzpy.intFloatListXyz.StrInListXyz`

  strCfg()
    get info

iradinapy.coloringIra module

simple tagging as '<color>' for simple coloring log messages on terminal(s) window or unix or ios using backend colorama. Using '<color>' because EZ human readable, So '<color>' are not supposed existing in log message. "{}".format() is not choosen because "{}" are present in log messages of contents of python dict (as JSON) etc.

Usage:
>> import iradinapy.coloringIra as COLS

Example:
>> log("this is in <green>color green<reset>, OK is in blue: <blue>OK?")

class iradinapy.coloringIra.**ColoringStream**
  Bases: `object`

  write my stream class only write and flush are used for the streaming https://docs.python.org/2/library/logging.handlers.html https://stackoverflow.com/questions/31999627/storing-logger-messages-in-a-string

  flush()

  write(*astr*)

iradinapy.coloringIra.**cleanColors**(*msg*)
  clean the message of color tags '<red> ...

iradinapy.coloringIra.**indent**(*msg, nb, car=' '*)
  indent nb car (spaces) multi lines message except first one

iradinapy.coloringIra.**log**(*msg*)
  elementary log stdout for debug if _verbose

iradinapy.coloringIra.**replace**(*msg, tags*)

iradinapy.coloringIra.**toColor**(*msg*)
  automatically clean the message of color tags '<red> ... if the terminal output stdout is redirected by user if not, replace tags with ansi color codes

iradinapy.coloringIra.**toColor_AnsiToWin32**(*msg*)
  for test debug no wrapping

iradinapy.configIra module

This file is the main API for config configparser for iradinaGUI

class `iradinapy.configIra.`**`ConfigManager`**(*runner*)

Bases: `object`

Manages the read/write of config files of iradinaGUI, and merges if useful.

file iradinaGUI_user.cfg

file iradinaGUI_default.cfg

`assertUserDefaultFiles`()

if inexisting, create config files user and default. relative to workdir

`checkFileExist`(*filename*)

filename as name relative to workdir

`getDefaultConfig`()

new instance default config

`getMainConfig`()

main as merged config of default plus overrides of user new instance main config

`getRealPath`(*name*)

`getUserConfig`()

new instance user config

`getWorkdir`()

`setMainConfig`(*cfg*)

set a user main config as global, is user choice

`iradinapy.configIra.`**`getCurrentMode`**()

`iradinapy.configIra.`**`getExistingModes`**()

`iradinapy.configIra.`**`getMainConfig`**()

`iradinapy.configIra.`**`isHidden`**(*item*, *nameAttr=None*, *modeName=None*)

avoid Components[*].Density because fnmatch pattern [seq] matches any character in seq. use Components*.Density instead

`iradinapy.configIra.`**`setCurrentMode`**(*modeName*)

iradinapy.controlSimulationIra module

class `iradinapy.controlSimulationIra.`**`ControlSimulationIra`**

Bases: `xyzpy.baseXyz._XyzConstrainBase`

general informations about

`setDefaultValues`()

not virtual, could be used

class `iradinapy.controlSimulationIra.`**`UserModeIra`**(*value=None*)

Bases: `xyzpy.intFloatListXyz.StrInListXyz`

iradinapy.controllerIra module

`iradinapy.controllerIra.`**`join`**`(`**v*`)`

as os.path.join but set antislash as slash, even for windows, keep windows 'c:'

`iradinapy.controllerIra.`**`launchFromSalomePyConsole`**`()`

iradinapy.dateTime module

This file contains DateTime and DeltaTime class

Usage:
>> import dateTime as DATT
>> ini = DATT.DateTime("now")
>> # some stuff
>> fin = DATT.DateTime("now")
>> duration = DATT.DeltaTime(ini, fin)

`class` `iradinapy.dateTime.`**`DateTime`**`(`*when=None*`)`

Bases: `object`

assume storing a date and hour, and conversions

Usage:
>> import dateTime as DATT
>> now = DATT.DateTime("now")
>> print("now is %s" % now)

`FORMAT_DATEHOUR_CONFIG = '%Y%m%d_%H%M%S'`

`FORMAT_DATE_CONFIG = '%Y%m%d'`

`FORMAT_FILE = '%Y%m%d_%H%M%S'`

`FORMAT_HOUR_CONFIG = '%H%M%S'`

`FORMAT_HUMAN = '%Y-%m-%d %H:%M:%S'`

`FORMAT_PACKAGE = '%Y-%m-%d %H:%M'`

`FORMAT_XML = '%Y/%m/%d %Hh%Mm%Ss'`

`MSG_UNDEFINED = 'UndefinedTime'`

`addSeconds``(`*secs*`)`

add seconds at time

`getSecondsToNow``()`

`getValue``()`

`isOk``()`

return True if ok

`localTime``()`

`raiseIfKo``()`

raise an exception with message why if not ok, else return self. This trick is to write usage

Usage:
>> aTimeOk = aTime.raiseIfKo() # raise Exception if KO
>> doSomethingWithaTimeOk(aTimeOk) # here i am sure that is OK

setValue (*time*)
choice as not deep copying if mutables value

toSeconds ()

toStrDateConfig ()

toStrDateHourConfig ()

toStrFile ()
use self.FORMAT_FILE, sortable, 2018-05-07… as '20180507_235958'

toStrHourConfig ()

toStrHuman ()
use self.FORMAT_HUMAN

toStrPackage ()

toStrXml ()

class `iradinapy.dateTime.`**DeltaTime** (*t1=None*, *t2=None*)
Bases: `object`

assume storing a duration, delta between two DateTime, and conversions

Usage:
>> import dateTime as DATT
>> t1 = DATT.DateTime("now")
>> time.sleep(3)
>> t2 = DATT.DateTime("now")
>> delta = DATT.DeltaTime(t1, t2)
>> print("delta time is %s" % delta)

MSG_UNDEFINED = 'UndefinedDeltaTime'

getT1 (*t*)

getT2 (*t*)

getValue ()
idem toSeconds()

isOk ()
return True if ok

raiseIfKo ()
raise an exception with message why if not ok, else return self. This trick is to write usage

Usage:
>> aDeltaTimeOk = adeltaTime.raiseIfKo() # raise Exception if KO
>> doSomethingWithaDeltaTimeOk(aDeltaTimeOk) # here i am sure that is OK

setT1 (*t*)

setT2 (*t*)

toMinutes()

toSeconds()

toStrHms()
all unities, hours and minutes and seconds as '2h34m56s'

toStrHuman()
automatic best unity, hours or minutes or seconds

iradinapy.dateTime.**date_to_datetime**(*date*)
From a string date as pyconf config.VARS.datehour 'YYYYMMDD_HHMMSS' returns [year, month, day, hour, minutes, seconds]

Parameters **date** – (str) The date in format YYYYMMDD_HHMMSS

Returns (tuple) as (str,str,str,str,str,str) The same date and time in separate variables.

iradinapy.dateTime.**fromDateHourConfig**(*datehour*)
datehour as pyconf config.VARS.datehour 'YYYYMMDD_HHMMSS'. Returns datetime.datetime

iradinapy.dateTime.**fromTimeStamp**(*val*)
Returns datetime.datetime

iradinapy.dateTime.**getWeekDayNow**()
Returns monday as 0, tuesday as 1 etc.

iradinapy.dateTime.**parse_date**(*date*)
Transform as pyconf config.VARS.datehour 'YYYYMMDD_HHMMSS' to 'YYYY-MM-DD hh:mm:ss'.

Parameters **date** – (str) The date to transform

Returns (str) The date in the new format

iradinapy.dateTime.**sleep**(*seconds*)
as time.sleep(seconds)

iradinapy.dateTime.**timedelta_total_seconds**(*timedelta*)
Replace total_seconds from datetime module in order to be compatible with old python versions

Parameters **timedelta** – (datetime.timedelta) The delta between two dates

Returns (float) The number of seconds corresponding to timedelta.

iradinapy.iradinaFilePatterns module

all file patterns of supposedly (sometimes) created by iradinaGui replaces '@xxx@' as file.in autotools

iradinapy.iradinaFilePatterns.**execReplaces**(*aStr*, *replaces=[]*)
append standart useful replaces to users replaces

iradinapy.iradinaFilePatterns.**filterColumsNamesForUranie**(*header*)
assume inexisting header array with empty strings '' lenght of #COLUMN_NAMES fill incomplete array with ''

iradinapy.iradinaFilePatterns.**filterHeaderDat**(*line*)
extract (key, value) from line from file .dat uranie/salome/paraview

iradinapy.iradinaFilePatterns.**getBaseFiles**()
list of useful mandatory files in uranie etude directory

iradinapy.iradinaFilePatterns.**getDefaultDivide**(*nb*)
Canvas.Divide by default

iradinapy.iradinaFilePatterns.**getFilePatterns**(*name*, *replaces=[]*)

iradinapy.iradinaFilePatterns.**getFixedUsefulDirs**(*rootDir*)
unconditionaly fixed useful

`iradinapy.iradinaFilePatterns.`**`getHeaderContentsFileDat`**`(`*filenameIni*`)`
header description of .dat uranie/salome/paraview returns a dict with keys NAME,DATE… message if inexisting file

`iradinapy.iradinaFilePatterns.`**`getHeaderFileDat`**`(`*filename*, *Verbose=False*`)`
header description of .dat uranie/salome/paraview returns a dict with keys #NAME, #DATE…

`iradinapy.iradinaFilePatterns.`**`getIgnoreFilesForCpack`**`(`*rootDir*, *useful*, *Verbose=False*`)`
list of usefless directories in uranie etude directory, not packaging saved

`iradinapy.iradinaFilePatterns.`**`getOtherUsefulDirsForCpack`**`(`*rootDir*`)`
unconditionaly fixed useful are not in other useful directories get user subdirs with CMakeLists.txt as useful

`iradinapy.iradinaFilePatterns.`**`getPatternKeys`**`()`

`iradinapy.iradinaFilePatterns.`**`getStdReplaces`**`()`
standart useful replaces, @xxx@ as .in autotools

`iradinapy.iradinaFilePatterns.`**`getTypesOfVisualize`**`()`

`iradinapy.iradinaFilePatterns.`**`getUsefulDirsForCpack`**`(`*rootDir*`)`

`iradinapy.iradinaFilePatterns.`**`getUselessDirsForCpack`**`(`*rootDir*`)`
list of useless directories in uranie etude directory, not packaging saved

`iradinapy.iradinaFilePatterns.`**`getUselessFilesForCpack`**`(`*rootDir*, *useful*`)`
list of usefless files in uranie etude directory, not packaging saved

`iradinapy.iradinaFilePatterns.`**`getVisualizeMethod`**`(`*aName*`)`

`iradinapy.iradinaFilePatterns.`**`isPresentCMakeLists`**`(`*aDir*`)`
test if CMakeLists.txt exists in directory

`iradinapy.iradinaFilePatterns.`**`isUranieColumn`**`(`*name*`)`

`iradinapy.iradinaFilePatterns.`**`listOnlyDirs`**`(`*rootDir*`)`
return list of expanded directories names in directory

`iradinapy.iradinaFilePatterns.`**`removeDuplicates`**`(`**args*`)`

`iradinapy.iradinaFilePatterns.`**`visualize_TDS_graphic`**`(`*FILEDAT*, *ATTS*, *EXPR=[]*, *DIVIDE=None*, *FILS=[]*, *OPTS=[]*, *Verbose=True*`)`
ATTS = ((“x1:x2”), (“x3:x5”)) for example 2 Draws

`iradinapy.iradinaFilePatterns.`**`visualize_TDS_univariate`**`(`*FILEDAT*, *ATTS*, *EXPR=[]*, *DIVIDE=None*, *FILS=[]*, *OPTS=[]*, *Verbose=True*`)`

iradinapy.iradinaGui module

This file is the main API file for iradinaGUI

Warning: NO ‘__main__ ‘ call allowed,
Use ‘../iradinaGUI’ (in parent directory)

Usage: see file ../iradinaGUI

class `iradinapy.iradinaGui.`**`ArgumentParserNoExit`**(*prog=None*, *usage=None*, *description=None*, *epilog=None*, *parents=[]*, *formatter_class=<class 'argparse.HelpFormatter'>*, *prefix_chars='-'*, *fromfile_prefix_chars=None*, *argument_default=None*, *conflict_handler='error'*, *add_help=True*, *allow_abbrev=True*)

Bases: `argparse.ArgumentParser`

change Exiting method as no exit

`exit`(*status=0*, *message=None*)

`filter_existing_file`(*string*)

`filter_float_positive`(*string*)

`filter_int_positive`(*string*)

`filter_list`(*string*)
parser filter from string 'xx,yy,zz,. . . ' returns list (if not error with python exec(value=[xx,yy,zz,. . .]))

`filter_list_float`(*string*)
parser filter from string 'xx,yy,zz,. . . ' returns list (if not error with python exec(value=[xx,yy,zz,. . .]))

`filter_list_int`(*string*)
parser filter from string 'xx,yy,zz,. . . ' returns list (if not error with python exec(value=[xx,yy,zz,. . .]))

`filter_logLevel`(*aStr*)

`filter_range`(*string*)
parser filter from string 'vmin,vmax' returns list (if not error with python exec(value=[xx,yy]))

`filter_square`(*string*)

`filter_workdir`(*string*)

`getLogLevels`()

`getLogLevelsStr`()

class `iradinapy.iradinaGui.`**`Ira`**(*logger*)
Bases: `object`

The main class that stores all the commands of iradinaGui (usually known as 'runner' argument in Command classes)

`assumeAsList`(*strOrList*)

`execute_cli`(*cli_arguments*)
select first argument as a command in directory 'commands', and launch on arguments

Parameters **`cli_arguments`** – (str or list) The iradinaGUI CLI arguments (as sys.argv)

`getAnswer`(*msg*)
question and user answer (in console) if confirm mode and not batch mode.

returns 'YES' or 'NO' if confirm mode and not batch mode

returns 'YES' if batch mode

`getBatchMode`()

`getColoredVersion`()
get colored iradinaGui version message

getConfig()

getConfirmMode()

getId()
assimiled as integer incremented on _idCommandHandlers

getLogger()

getOptions()

get_help()
get general help colored string

parseArguments(*arguments*)

print_help()
prints iradinaGui general help

runGUI()
a main window for iradinaGUI

setConfirmMode(*value*)

show_doc()
show iradinaGui general documentation

iradinapy.iradinaGui.**assumeAsList**(*strOrList*)
return a list as sys.argv if string

iradinapy.iradinaGui.**cmdsdir = '/volatile2/wambeke/TULEAP_MATIX/MATIX_26-C07/SOURCES/IRA**

if DBG.isDeveloper(): workdirdefault = os.path.realpath(os.path.join(rootdir, "..", "IRADINAGUI_WORKDIR"))

else: workdirdefault = os.path.expandvars(os.path.join("$HOME", "IRADINAGUI_WORKDIR"))

iradinapy.iradinaGui.**getVersion**()
get version number as string

iradinapy.iradinaGui.**launchIra**(*command*)
launch iradinaGUI as subprocess.Popen command as string ('iradinaGUI –help' for example) used for unittest, or else…

Returns RCO.ReturnCode

iradinapy.iradinaSettings module

class iradinapy.iradinaSettings.**IradinaSettings**
Bases: *settingspy.settings.Settings* (page 72)

may be future link to Qt QSettings for future do not forget window environment variables are NOT case sensitive user have to write environment variables as syntax ${…}

policy: names setting variables beginning with "_" contains environment variables reference

iradinapy.iradinaSettings.**checkAll**()
used as singleton

iradinapy.iradinaSettings.**checkEnvVar**(*val*)
used as singleton

iradinapy.iradinaSettings.**getExpandedVar**(*name*)
used as singleton

iradinapy.iradinaSettings.**getIradinaSysMacrosDir**()

iradinapy.iradinaSettings.**getSettings**()
used as singleton

iradinapy.iradinaSettings.**getVar**(*name*)
    used as singleton

iradinapy.iradinaSettings.**setEnvVar**(*envVar*, *value*)

iradinapy.iradinaSettings.**setEnvVarByDefault**(*envVar*, *valueDefault*)

iradinapy.loggingIra module

iradinaGui logger. using logging package

Define one logger with one handler on stdout and one handler on file for production.
Define another one logger for unittest.

see: http://sametmax.com/ecrire-des-logs-en-python

Define two LoggerIra instances in iradinaGui, no more need.
    - _loggerDefault as production/development logger
    - _loggerUnittest as unittest logger

see use of handlers of _loggerDefault for
log console and log files xml, txt

console handler:
    - info() : no format
    - error() warning() trace() debug() etc. :
        formatted indented on multi lines messages using handlers

file handlers:
    - info() error() warning() trace() debug() etc. :
        formatted indented on multi lines messages using handlers

WARNING:
    log step and log trace are present on stdout console or log file
    following level handlers settings

class iradinapy.loggingIra.**DefaultFormatter**(*fmt=None*, *datefmt=None*, *style='%'*)
    Bases: logging.Formatter

    format(*record*)
        Format the specified record as text.

        The record's attribute dictionary is used as the operand to a string formatting operation which yields the returned string. Before formatting the dictionary, a couple of preparatory steps are carried out. The message attribute of the record is computed using LogRecord.getMessage(). If the formatting string uses the time (as determined by a call to usesTime(), formatTime() is called to format the event time. If there is exception information, it is formatted using formatException() and appended to the message.

    setColorLevelname(*levelname*)
        set color implies color special characters and tabulate levelname length of string

class iradinapy.loggingIra.**FileTxtFormatter**(*fmt=None*, *datefmt=None*, *style='%'*)
    Bases: logging.Formatter

format (*record*)

Format the specified record as text.

The record's attribute dictionary is used as the operand to a string formatting operation which yields the returned string. Before formatting the dictionary, a couple of preparatory steps are carried out. The message attribute of the record is computed using LogRecord.getMessage(). If the formatting string uses the time (as determined by a call to usesTime(), formatTime() is called to format the event time. If there is exception information, it is formatted using formatException() and appended to the message.

class iradinapy.loggingIra.**FileXmlFormatter** (*fmt=None*, *datefmt=None*, *style='%'*)

Bases: `logging.Formatter`

format (*record*)

Format the specified record as text.

The record's attribute dictionary is used as the operand to a string formatting operation which yields the returned string. Before formatting the dictionary, a couple of preparatory steps are carried out. The message attribute of the record is computed using LogRecord.getMessage(). If the formatting string uses the time (as determined by a call to usesTime(), formatTime() is called to format the event time. If there is exception information, it is formatted using formatException() and appended to the message.

class iradinapy.loggingIra.**LoggerIra** (*name*, *level=20*)

Bases: `logging.Logger`

Inherited class logging.Logger for logger iradinaGui

add a level STEP as log.step(msg)
add a level TRACE as log.trace(msg)
below log.info(msg)
above log.debug(msg)
to assume message step inside files xml 'command's internal traces'
to assume store long log asci in files txt outside files xml

see: /usr/lib64/python2.7/logging/__init__.py etc.

close ()

final stuff for logger, done at end iradinaGui flushed and closed xml files have to be not overriden/appended

closeFileHandlerForCommand (*cmdInstance*)

getMainCommandHandler ()

returns handler for colored stdout console/terminal for human user eye iradinaGUI outputs

initLinkForCommand (*cmdParent*, *cmdNew*)

logStep (*step*)

current logger.info() step as 'header … etc … '

logStep_begin (*header*, *step="*)

initialize for main handler (tty as stdout) a one line message for steps (…of compilation for example) as no return line message with logger.info() level

example:
'header … first temporary step message … '
'header … etc … ' (on same line)
'header … OK' (on same line)

logStep_end(*step*, *tab=None*)
last logger.info() step as 'header … OK' or 'header … KO'

setFileHandlerForCommand(*cmdParent*, *cmdInstance*)
add file handler to logger to set log files for a iradinaGui command. when command is known from pyconf/config instance

Example:
log files names for command prepare
with micro commands clean/source/patch
~/LOGS/20180510_140606_prepare_lenovo.xml
~/LOGS/OUT/20180510_140606_prepare_lenovo.txt
~/LOGS/micro_20180510_140607_clean_lenovo.xml
~/LOGS/OUT/micro_20180510_140607_clean_lenovo.txt
etc.

setLevelMainHandler(*level*)

step(*msg*, **args*, ***kwargs*)
Log 'msg % args' with severity '_STEP'.

testNoReturn()
test when message ending '…' and level info then no return mode

trace(*msg*, **args*, ***kwargs*)
Log 'msg % args' with severity '_TRACE'.

xx_isEnabledFor(*level*)
Is this logger enabled for level 'level'? currently not modified from logging.Logger class, here only for call log debug.

class `iradinapy.loggingIra.`**StreamHandlerIra**(*stream=None*)
Bases: `logging.StreamHandler`

A handler class which writes logging records, appropriately formatted, to a stream. Note that this class does not close the stream, as sys.stdout or sys.stderr may be used.

from logging.StreamHandler class, modified for 'no return' mode line if '…' at end of record message

emit(*record*)
Emit a record.

If a formatter is specified, it is used to format the record. The record is then written to the stream with a trailing newline. If exception information is present, it is formatted using traceback.print_exception and appended to the stream. If the stream has an 'encoding' attribute, it is used to determine how to do the output to the stream.

isLastRecordHaveNoReturn()
to memorize if last info record is 'no return' mode (as ending '…') avoid define inherited __init__

isNeedFirstReturn(*record*)
'no return' mode valid only if 2 consecutives info messages if not, needs insert return line BEFORE (warning, debug, or other) current record message

class `iradinapy.loggingIra.`**UnittestFormatter**(*fmt=None*, *datefmt=None*, *style='%'*)
Bases: `logging.Formatter`

format(*record*)
Format the specified record as text.

The record's attribute dictionary is used as the operand to a string formatting operation which yields the returned string. Before formatting the dictionary, a couple of preparatory steps are carried out. The message attribute of the record is computed using LogRecord.getMessage(). If the formatting string uses the time (as determined by a call to usesTime(), formatTime() is called to format the event time. If there is exception information, it is formatted using formatException() and appended to the message.

class `iradinapy.loggingIra.`**`UnittestStream`**

Bases: `object`

write my stream class only write and flush are used for the streaming

https://docs.python.org/2/library/logging.handlers.html

https://stackoverflow.com/questions/31999627/storing-logger-messages-in-a-string

`flush``()`

`getLogs``()`

`getLogsAndClear``()`

`write``(`*astr*`)`

final method called when message is logged

class `iradinapy.loggingIra.`**`XmlHandler`**`(`*capacity*`)`

Bases: `logging.handlers.BufferingHandler`

log outputs in memory as BufferingHandler. Write ElementTree in file and flush are done once when method close is called, to generate xml file.

ats = {
“fileName”: xml file name of micro command
“command”: cmd, # ‘compile’ or ‘prepare’ etc.
“passed”: res, # ‘O’ or ‘1’
“launchedCommand” : fullcmd, # ‘compile TOTO -etc’
}

see: https://docs.python.org/2/library/logging.handlers.html

`close``()`

prepare ElementTree from existing logs and write xml file

warning: avoid iradinaGUI logging message in logger close phase

`createLogField``()`

prepare formatted string from self.buffer LogRecord for xml ‘Log’ node using handler formatter

`createLogFieldFromScrath``()`

prepare formatted string from self.buffer LogRecord for xml ‘Log’ node local format

`set_config``(`*config*`)`

config is supposedly non existing, no overwrite accepted

`set_target_file``(`*filename*`)`

filename is file name xml with path supposedly non existing, no overwrite accepted

`iradinapy.loggingIra.`**`filterLevel`**`(`*aLevel*`)`

filter levels logging values from firsts characters levels. No case sensitive.

example:
‘i’ -> ‘INFO’

‘cRiT’ -> ‘CRITICAL’

`iradinapy.loggingIra.`**`getCurrentLogger`**`()`
: get one of _loggerDefault or _loggerUnittest as first created by getDefautLogger() or getUnittestLogger():

`iradinapy.loggingIra.`**`getDefaultLogger`**`()`
: official method to get the only one instance of Default Logger

`iradinapy.loggingIra.`**`getListOfStrLogRecord`**`(`*listOfLogRecord*`)`
: Returns one line string for logging LogRecord description

`iradinapy.loggingIra.`**`getLogger`**`()`

`iradinapy.loggingIra.`**`getMessage`**`(`*self*`)`
: modified from logging.__init__.LogRecord.getMessage, better message on format error Return the message for this LogRecord.

  Return the message for this LogRecord after merging any user-supplied arguments with the message.

`iradinapy.loggingIra.`**`getStrDirLogger`**`(`*logger*`)`
: Returns multi line string for logger description, with dir(logger). Used for debug

`iradinapy.loggingIra.`**`getStrHandler`**`(`*handler*`)`
: Returns one line string for handler description (as inexisting __repr__) to avoid create inherited classe(s) handler

`iradinapy.loggingIra.`**`getStrLogRecord`**`(`*logRecord*`)`
: Returns one line string for simple logging LogRecord description

`iradinapy.loggingIra.`**`getStrShort`**`(`*msg*`)`
: Returns short string for msg (as first caracters without line feed

`iradinapy.loggingIra.`**`getUnittestLogger`**`()`
: official method to get the only one instance of Unittest Logger

`iradinapy.loggingIra.`**`indent`**`(`*msg*, *nb*, *car=' '*`)`
: indent nb car (spaces) multi lines message except first one

`iradinapy.loggingIra.`**`indentUnittest`**`(`*msg*, *prefix='| '*`)`
: indent multi lines message except first one with prefix. prefix default is designed for less spaces for size logs files and keep logs human eye readable

`iradinapy.loggingIra.`**`initLoggerAsDefault`**`(`*logger*, *fmt=None*, *level=None*`)`
: init logger as prefixed message and indented message if multi line exept info() outed ‘as it’ without any format. level could be modified during execution

`iradinapy.loggingIra.`**`initLoggerAsUnittest`**`(`*logger*, *fmt=None*, *level=None*`)`
: init logger as silent on stdout/stderr used for retrieve messages in memory for post execution unittest https://docs.python.org/2/library/logging.handlers.html

`iradinapy.loggingIra.`**`log`**`(`*msg*, *force=False*`)`
: elementary log when no logging.Logger yet

`iradinapy.loggingIra.`**`testLogger_1`**`(`*logger*`)`
: small test

`iradinapy.loggingIra.`**`testMain`**`()`

`iradinapy.loggingIra.`**`testNoReturn`**`(`*logger*`)`
: test when message ending ‘. . . ’ and level info then no return mode

iradinapy.mainWindowIra module

iradinapy.modelIra module

class iradinapy.modelIra.**ModelIra**

Bases: `xyzpy.baseXyz._XyzConstrainBase`

general instance to group all iradina data files in a directory input file iradina release ?

getActionsContextMenu()

getEtudeWorkdirBrut(*expanded=True*)

getEtudeWorkdirExpanded()

getHistoryFile()

isHidden(*nameAttr*)

to know if attribute is currently displayed in treeView and other dialog widget

setDefaultValues()

not virtual, could be used

setFromFileIra(*fileName*, *verbose=False*)

override inherited method

toStrIra()

override inherited method

userExpand()

filename patterns '*,??' warning not for '[' ']' as 'alist[*]' no found way to quote meta-character https://docs.python.org/2/library/fnmatch.html

userMode()

change user mode

iradinapy.testPrerequisitesIra module

iradinapy.testPrerequisitesIra.**TestImports**()

test of prerequisites import python for IradinaGui message for problem(s), aborting immediatly.

iradinapy.testPrerequisitesIra.**TestIradinaFeatures**()

test of iradina CODE features

iradinapy.testPrerequisitesIra.**error**(*message*)

iradinapy.treeViewIra module

iradinapy.treeViewIra.**verboseEvent = False**

cosmetic stuff for treeView Iradina

iradinapy.utilsIra module

utilities for iradinaGUI general useful simple methods all-in-one import iradinapy.utilsIra as UTS

Usage:
>> import iradinapy.utilsIra as UTS
>> UTS.ensure_path_exists(path)

`iradinapy.utilsIra.`**`Popen`**(*command*, *shell=True*, *cwd=None*, *env=None*, *stdout=- 1*, *stderr=- 1*, *logger=None*)
make subprocess.Popen(cmd), with call logger.trace and logger.error if problem as returncode != 0

`iradinapy.utilsIra.`**`addSpaces`**(*idx*, *aStr*)

`iradinapy.utilsIra.`**`black`**(*msg*)

`iradinapy.utilsIra.`**`blue`**(*msg*)

`iradinapy.utilsIra.`**`critical`**(*msg*)

`iradinapy.utilsIra.`**`cyan`**(*msg*)

`iradinapy.utilsIra.`**`deepcopy_list`**(*input_list*)
Do a deep copy of a list

Parameters **`input_list`** – (list) The list to copy

Returns (list) The copy of the list

`iradinapy.utilsIra.`**`ensure_file_exists`**(*aFile*, *aDefaultFile*)
Create a file if not existing, copying from default file

Parameters

- **`aFilepath`** – (str) The file to ensure existence
- **`aDefaultFile`** – (str) The default file to copy if not existing

`iradinapy.utilsIra.`**`ensure_path_exists`**(*path*)
Create a path if not existing

Parameters **`path`** – (str) The path.

`iradinapy.utilsIra.`**`error`**(*msg*)

`iradinapy.utilsIra.`**`formatTuples`**(*tuples*)
Format 'label = value' the tuples in a tabulated way.

Parameters **`tuples`** – (list) The list of tuples to format

Returns (str) The tabulated text. (as mutiples lines)

`iradinapy.utilsIra.`**`formatValue`**(*label*, *value*, *suffix=''*)
format 'label = value' with the info color

Parameters

- **`label`** – (int) the label to print.
- **`value`** – (str) the value to print.
- **`suffix`** – (str) the optionnal suffix to add at the end.

`iradinapy.utilsIra.`**`get_iradinaGUI_version`**(*config*)

`iradinapy.utilsIra.`**`get_tmp_filename`**(*config*, *name*)

`iradinapy.utilsIra.`**`green`**(*msg*)

`iradinapy.utilsIra.`**`header`**(*msg*)

`iradinapy.utilsIra.`**`info`**(*msg*)

`iradinapy.utilsIra.`**`label`**(*msg*)

`iradinapy.utilsIra.`**`magenta`**(*msg*)

`iradinapy.utilsIra.`**`merge_dicts`**(**dict_args*)
Given any number of dicts, shallow copy and merge into a new dict, precedence goes to key value pairs in latter dicts.

`iradinapy.utilsIra.`**`normal`**(*msg*)

`iradinapy.utilsIra.`**`red`**(*msg*)

`iradinapy.utilsIra.`**`remove_item_from_list`**(*input_list*, *item*)
Remove all occurences of item from input_list

Parameters **`input_list`** – (list) The list to modify

Returns (list) The without any item

`iradinapy.utilsIra.`**`replace_in_file`**(*file_in*, *str_in*, *str_out*)
Replace <str_in> by <str_out> in file <file_in>. save a file old version as file_in + '_old'

Parameters

- **`file_in`** – (str) The file name
- **`str_in`** – (str) The string to search
- **`str_out`** – (str) The string to replace.

`iradinapy.utilsIra.`**`reset`**(*msg*)

`iradinapy.utilsIra.`**`sleep`**(*sec*)

`iradinapy.utilsIra.`**`success`**(*msg*)

`iradinapy.utilsIra.`**`tabColor`**(**args*)
return tabulated colored string from args, assume true length of color tags as <OK> <info> etc. to correct alignment when tags are interpreted as (no-length-spacing) color for colorama use or else

`iradinapy.utilsIra.`**`warning`**(*msg*)

`iradinapy.utilsIra.`**`white`**(*msg*)

`iradinapy.utilsIra.`**`yellow`**(*msg*)

Module contents

4.1.2 configparserpy

configparserpy package

Subpackages

configparserpy.test package

Submodules

configparserpy.test.test_130_configParserUtils module

utilities for best use ConfigParser

see:
https://wiki.python.org/moin/ConfigParserExamples
https://docs.python.org/2/library/configparser.html

class `configparserpy.test.test_130_configParserUtils.`**`TestCase`**(*methodName='runTest'*)

Bases: `unittest.case.TestCase`

Test the configParserUtils.py

`test_000`()

`test_010`()

`test_020`()

`test_030`()

`test_032`()

`test_034`()

`test_100`()

`test_999`()

Module contents

Submodules

configparserpy.configParserUtils module

utilities for best use ConfigParser

see: https://wiki.python.org/moin/ConfigParserExamples

class `configparserpy.configParserUtils.`**`UtSafeConfigParser`**(**args*, ***kwargs*)

Bases: `configparser.SafeConfigParser`

SafeConfigParser with ExtendedInterpolation, and __repr__, and readDefaultAndUser to merge default and user config

`copy`()

`isEmpty`()

`readDefaultAndUser`(*aStrDefault*, *aStrUser*)
merge user overriding origin defaults

`readFromStr`(*aStr*, *merge=False*)
allow merge only explicitly

`toCatchAll`(*verbose=False*)
permits class attribute writings, (but raise on accentuation and avoid spaces in section names)

cfg = UtSafeConfigParser()
cfg.readFromStr(‘” | [General]
reporter = tintin
‘”)
config = cfg.toCatchAll()
print(config.General.reporter) # -> “tintin”

toDict(*verbose=False*)

toDictTuple(*verbose=False*)

toOrderedDict(*verbose=False*)

writeToStr()
allow merge only explicitly

`configparserpy.configParserUtils.`**getConfigFromDefaultAndUserStr**(*aStrDefault*, *aStrUser*)
simple create config from Default an overrrides from User as strings

`configparserpy.configParserUtils.`**getConfigFromFile**(*aFile*)

`configparserpy.configParserUtils.`**getConfigFromStr**(*aStr*)
simple create config from contents as string

Module contents

4.1.3 settingspy

settingspy package

Submodules

settingspy.setStyleFactory module

`settingspy.setStyleFactory.`**run**()

settingspy.settings module

class `settingspy.settings.`**Settings**
Bases: `object`

may be future link to Qt QSettings for future do not forget window environment variables are NOT case sensitive user have to write environment variables as syntax ${. . . }

policy: names setting variables beginning with "_" contains environment variables reference

checkAll()
expand var in settings return (ok, aDict) ok is False or True aDict is settings as aDict[key] = (value, interpretedValue)

checkEnvVar(*aStrWithEnvVar*)
check all env vars contained as syntax ${. . . } defined in environ

getExpandedVar(*name*)
returns expanded value of var name, or None if inexisting or problem in expand

getRealPath(*aPathWithEnvVar*)
resolve file path env variable as ${HOME}/toto etc. . . with os.path.expandvars interpretation of env var

getVar(*name*, *Verbose=True*)
return NOT expanded value of var name

setVar(*name*, *value*, *Verbose=True*)
no control, user choice with checkAll()

`settingspy.settings.`**getExpandedVar**(*name*)
used as singleton

`settingspy.settings.`**`getSettings`**`()`
used as singleton

`settingspy.settings.`**`getVar`**`(`*name*`)`
used as singleton

`settingspy.settings.`**`setEnvVar`**`(`*envVar*, *value*`)`
with message warning if change

`settingspy.settings.`**`setEnvVarByDefault`**`(`*envVar*, *valueDefault*`)`
only for single environ variables: envVar='HOME' for ${HOME} or $HOME valueDefault have to be expanded (i.e. without '$')

Module contents

settingspy for salome matix

4.1.4 filewatcherpy

filewatcherpy package

Subpackages

filewatcherpy.test package

Submodules

filewatcherpy.test.test_340_fileWatcher module

`class` `filewatcherpy.test.test_340_fileWatcher.`**`TestCase`**`(`*methodName='runTest'*`)`
Bases: `unittest.case.TestCase`

`test_010``()`

`test_020``()`

`test_030``()`

`test_999``()`

Module contents

Submodules

filewatcherpy.fileWatcher module

`filewatcherpy.fileWatcher.`**`exampleLaunchStandalone`**`()`

`filewatcherpy.fileWatcher.`**`getFileWatcher`**`()`
use it as singleton

`filewatcherpy.fileWatcher.`**`getRealPath`**`(`*aPathWithEnvVar*`)`
resolve env variable as $HOME/toto etc… with expandvars

`filewatcherpy.fileWatcher.`**`getRealPath_obsolete`**`(`*aPathWithEnvVar*`)`
resolve env variable as $HOME/toto etc… with subprocess shell interpretation of env var

Module contents

filewatcherpy for salome matix

CHAPTER FIVE

RELEASE NOTES

5.1 Release notes

In construction.

PYTHON MODULE INDEX

INDEX