

SyncPy2 Method Wizard UI documentation

Part 1 – Installation	2
Part 2 - SyncPy2 Method Wizard UI	3
1. Create a method from scratch	3
a. Choose the input data	3
b. Set the main information	4
c. Set the arguments	4
d. Generate and save the method.....	5
2. Import an existing method.....	5

Part 1 – Installation

You need to install anaconda to be able to launch the entire SyncPy suite. If you download the SyncPy installer from the website, it will also install anaconda for you.

You can access the UIs either from the src folder inside the SyncPy installation folder or shortcuts on your desktop if you are on Windows/Mac, or in your home folder if you are on Linux, if you installed SyncPy with the installer.

Part 2 - SyncPy2 Method Wizard UI

The purpose of this tool is to help to create, from scratch or an example, python coder to implement new Methods that the interface can use to process data.

Syncpy method wizard UI

About

Method

Input Data types

Univariate Categorical Linear

Number of input signals : 1 Import from existing Methods... 1

Name Author

Description

Arguments

Save new

Name Default value False add 2

Type bool Hint

Name	Type	Default value	Hint
------	------	---------------	------

Generate .py 3

Save... 4

SyncPy Method Wizard (v 2.0) successfully loaded

1. Create a method from scratch

a. Choose the input data

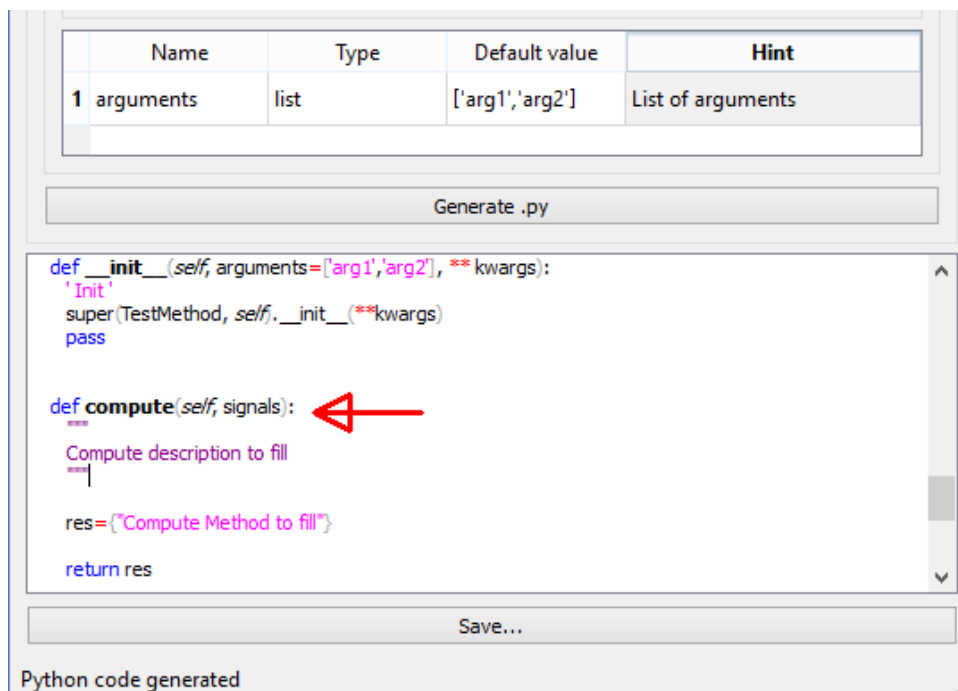
First you have to choose on which kind of data the method will work. In the *Input Data types* panel choose the types of the signals, and the number of signals to process.

Input Data types

Univariate Categorical Linear

Number of input signals : 1 Import from existing Methods...

Or you can use the button to import an existing Method and it will deduce the types for you (see [section do to what you want to do](#)).



Warning: generating the python file will not save it! To do it click on the *Save* button (4), it will automatically suggest you to save the python file in the right place to be able to use it in the SyncPy UI.

[Import an existing method](#) section).

b. Set the main information

Then you have to choose a name for your method. This will be the name of the generated file and the name displayed in the SyncPy UI. Set the name of the author and a description of what your method is doing.

c. Set the arguments

Arguments

Save new

Name Default value

Type Hint

Name	Type	Default value	Hint

Now you can start to add the arguments of your method. For each argument you **must** specify a name and a default value. For each type of argument, a default value is set to show you the format you must enter. **Warning: if you change the type of the argument you will lose the default value you set!**

If an argument with the same name is already added you can't add it. The hint is optional but useful for the community. Click on the *add* button (2) to add your argument in the arguments list. You can erase at any moment an argument in the list by right clicking on it and choose *delete row*.

Arguments

Save new

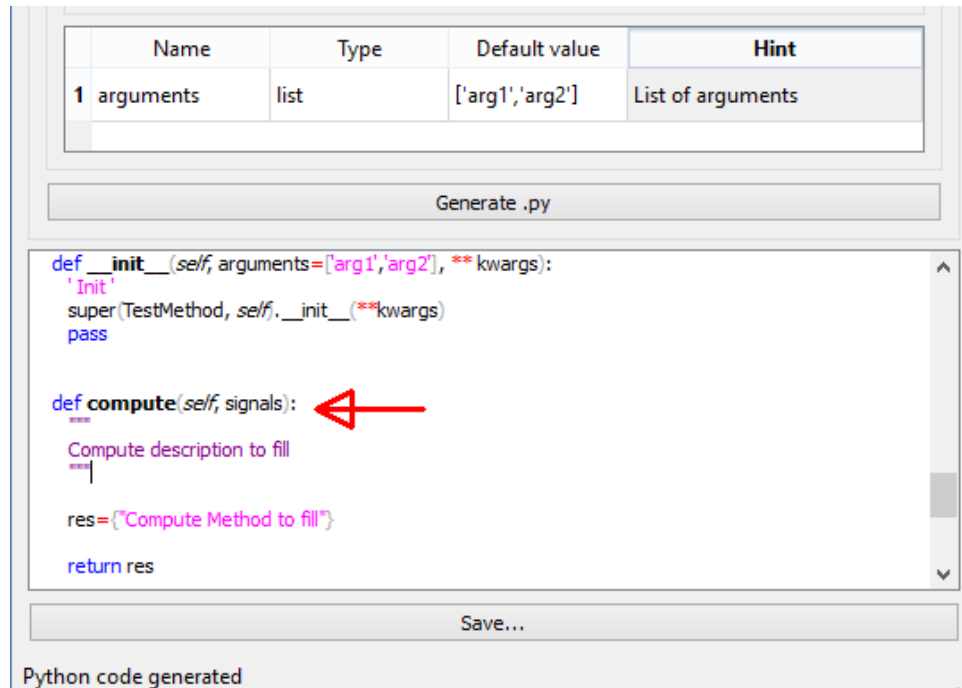
Name Default value

Type Hint

	Name	Type	Default value	Hint	
1	arguments	list	['arg1','arg2']	List of arguments	<input type="button" value="delete row"/>

d. Generate and save the method

Clicking on the *Generate .py* button (3), the wizard will create your method with all the information you gave. The final step is to edit the *compute* function do to what you want to do.



The screenshot shows a wizard interface. At the top, there is a table with the following data:

	Name	Type	Default value	Hint
1	arguments	list	['arg1','arg2']	List of arguments

Below the table is a button labeled "Generate .py". Underneath the button is a text area containing the following Python code:

```
def __init__(self, arguments=['arg1','arg2'], **kwargs):  
    'Init'  
    super(TestMethod, self).__init__(**kwargs)  
    pass  
  
def compute(self, signals):  
    """  
    Compute description to fill  
    """  
    res = {"Compute Method to fill"}  
    return res
```

A red arrow points to the `compute` function definition. Below the code area is a button labeled "Save...". At the bottom of the window, it says "Python code generated".

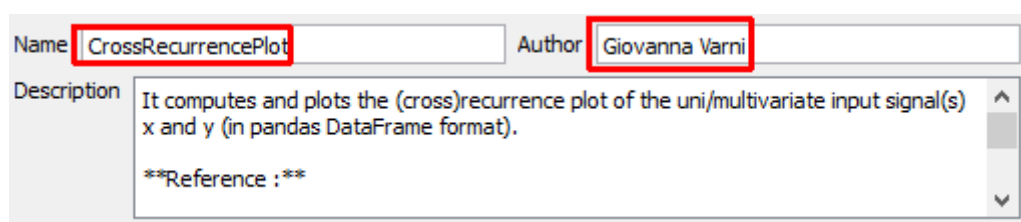
Warning: generating the python file will not save it! To do it click on the *Save* button (4), it will automatically suggest you to save the python file in the right place to be able to use it in the SyncPy UI.

2. Import an existing method

You can also import an existing method by clicking on the *Import from existing Methods* button (1). Before that you can set the type of the signals and the number of signals, the wizard will only show methods working on the same signals you want to process.

The wizard will fill all the information, add the arguments to the argument list and generate the python file. **Warning: you will have to rewrite the compute method, the wizard will not copy it.**

Be careful to **change the name of your method**, generate the python file and modify the compute function before saving the method.



The screenshot shows a wizard interface for importing a method. The "Name" field is filled with "CrossRecurrencePlot" and the "Author" field is filled with "Giovanna Varni". Both fields are highlighted with red boxes. Below these fields is a "Description" text area containing the following text:

It computes and plots the (cross)recurrence plot of the uni/multivariate input signal(s) x and y (in pandas DataFrame format).

**Reference : **