CONTENTS

Igor Pro – Python プログラミングの基礎 / Python programming basics

はじめに / Introduction

Igor Pro v10.0 より、Igor Pro 内で Python のプログラミングを行えるようになりました。/ As of Igor Pro v10.0, it is now possible to program in Python within Igor Pro.
Python プログラムから Igor Pro 内のオブジェクトをコントロールできるだけでなく、Igor Pro のプログラムから Python のコードを呼び出すことができます。/ Not only can you control objects in Igor Pro from Python programs, but you can also call Python code from Igor Pro programs.

これにより、Python からアクセスできる外部のライブラリ、例えば、機械学習のライブラリと Igor Pro を連動させることができるようになります。/ This will allow you to link Igor Pro with external libraries that can be accessed from Python, such as machine learning libraries.

ここでは、Igor Pro プログラム側と Python プログラム側の基本的なコマンドを列挙します。/ Here, we will list the basic commands for the Igor Pro program and Python program.

Python 環境の設定方法は、ヘルプ Python.ifn を参照してください。/ For information on how to set up the Python environment, please refer to the Help “Python.ifn”.

Python → Igor Pro / Python -> Igor Pro

以下は、Python メニューで開く、Python Console で行います。/ The followings are done in the Python Console, which is opened by Python menu.

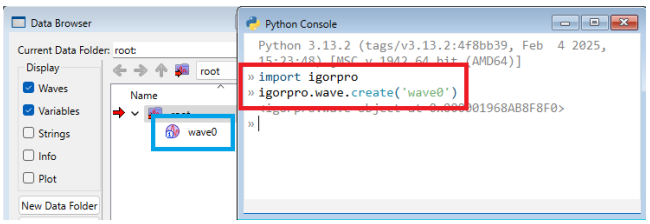
Igor オブジェクトの作成 / Creating Igor objects

1D ウェーブの作成 / Creating a 1D wave

[Python]

```
igorpro.wave.create('wave0')
```

wave0 という 1D ウェーブが作成されます。/ A 1D wave called “wave0” is created.

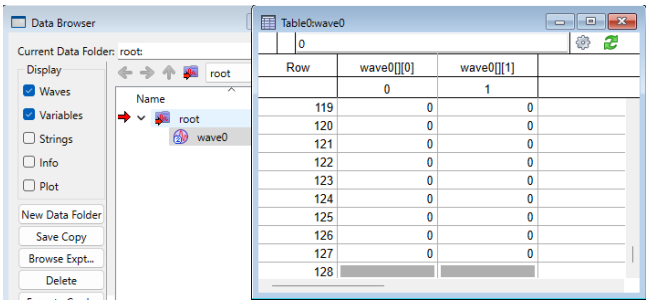


2D ウェーブの作成 / Creating a 2D wave

[Python]

```
igorpro.wave.create('wave0',shape=(128,2))
```

wave0 という 2D ウェーブが作成されます。/ A 2D wave called “wave0” is created.

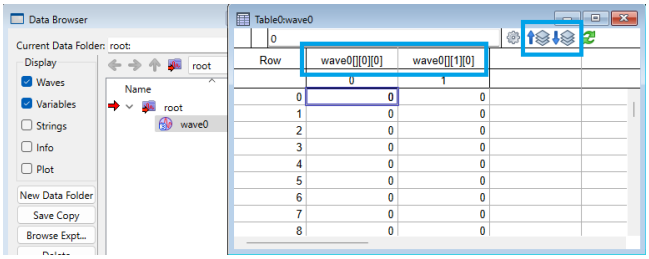


3D ウェーブの作成 / Creating a 3D wave

[Python]

```
igorpro.wave.create('wave0',shape=(128,2,2)
)
```

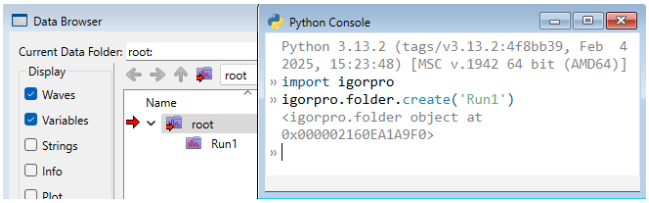
wave0 という 3D ウェーブが作成されます。/ A 3D wave called “wave0” is created.



データフォルダーの作成 / Creating a data folder

[Python]

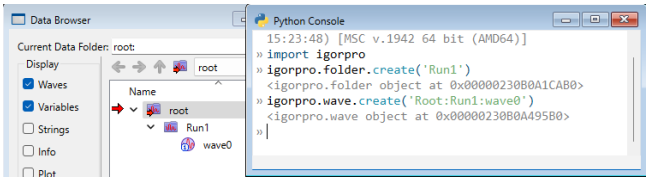
```
igorpro.folder.create('Run1')
```



データフォルダー（Run1）内にウェーブを作成 / Creating a wave in a data folder ‘Run1’

[Python]

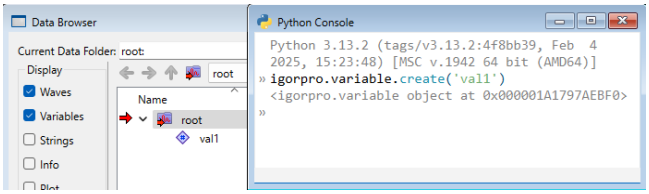
```
igorpro.wave.create('Root:Run1:wave0')
```



変数の作成 / Creating a variable

[Python]

```
igorpro.variable.create('val1')
```

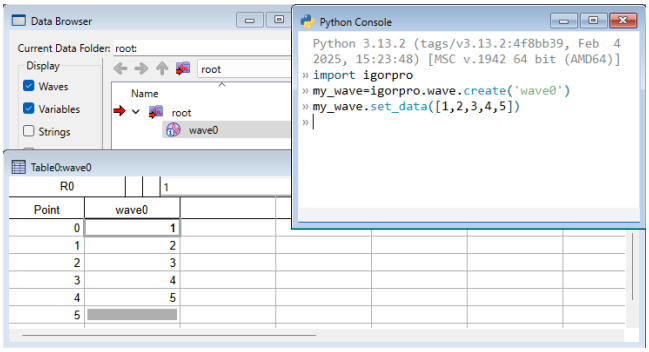


Igor オブジェクトへのデータの代入 / Assigning data to Igor objects

1D ウェーブへの代入（配列） / Assignment to a 1D wave (Array)

[Python]

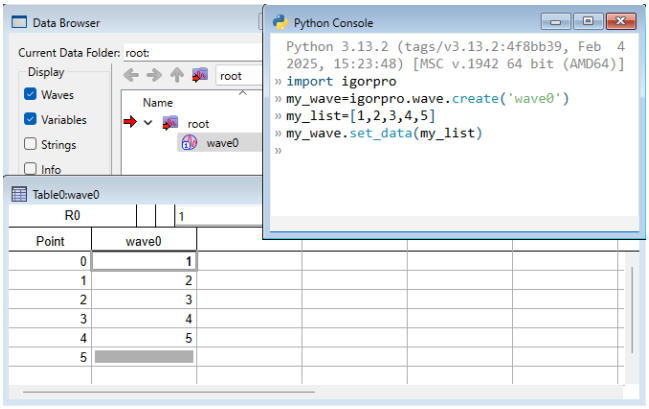
```
my_wave=igorpro.wave.create('wave0')
my_wave.set_data([1,2,3,4,5])
```



1D ウェーブへの代入（変数） / Assignment to a 1D wave (Variable)

[Python]

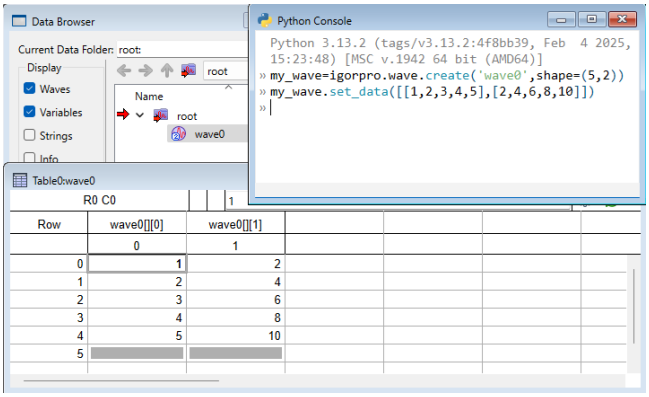
```
my_wave=igorpro.wave.create('wave0')
my_list=[1,2,3,4,5]
my_wave.set_data(my_list)
```



2D ウェーブへの代入（行列） / Assignment to a 2D wave (Matrix)

[Python]

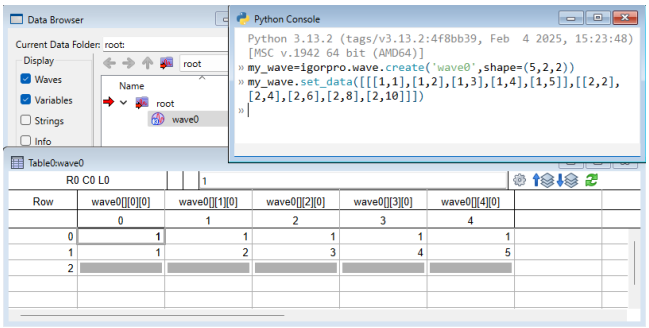
```
my_wave=igorpro.wave.create('wave0',shape=(5,2))
my_wave.set_data([[1,2,3,4,5],[2,4,6,8,10]])
```



3D ウェーブへの代入（行列） / Assignment to a 3D wave (Matrix)

[Python]

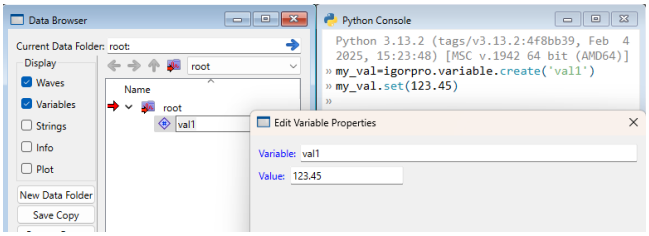
```
my_wave=igorpro.wave.create('wave0',shape=(5,2,2))
my_wave.set_data([
        [[1,1],[1,2],[1,3],[1,4],[1,5]],
        [[2,2],[2,4],[2,6],[2,8],[2,10]]])
```



変数への値の代入 / Assignment to a variable

[Python]

```
my_val=igorpro.variable.create('val1')
my_val.set(123.45)
```

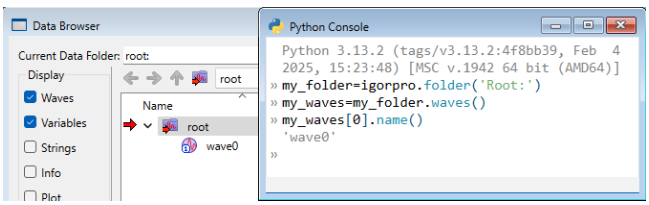


Igor オブジェクトからのデータの取得 / Getting data from Igor objects

Igor Pro 内のウェーブの名前をインデックスで取得 / Get a name of waves in Igor Pro by index

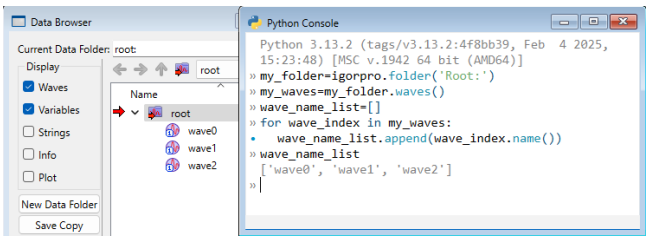
[Python]

```
my_folder=igorpro.folder('Root:')
my_waves=my_folder.waves()
my_waves[0].name()
```



ウェーブのリストを処理するには `folder` を経由して行います。/ To process the list of waves, go through the `folder`.

Igor Pro 内のウェーブ名のリストを取得 / Get a list of wave names in Igor Pro



[Python]

```
my_folder=igorpro.folder('Root:')
my_waves=my_folder.waves()
wave_name_list=[]
for wave_index in my_waves:
   wave_name_list.append(wave_index.name())
```

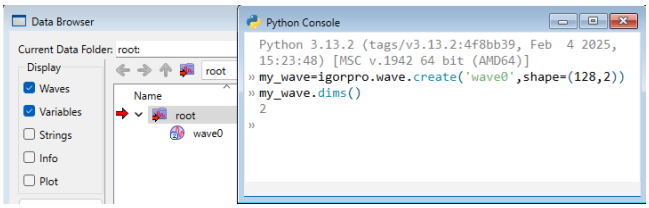
リストの内容は次のようになります。/ The contents of the list are as follows.

[Python]

```
wave_name_list
```

```
['wave0', 'wave1', 'wave2']
```

ウェーブの次元の数を取得 / Get the number of wave dimensions

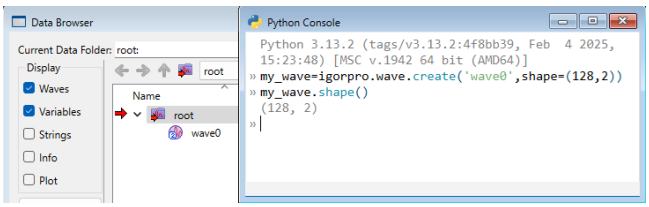


[Python]

```
my_wave=igorpro.wave.create('wave0',shape=(128,2))
my_wave.dims()
```

```
2
```

ウェーブの次元の形を取得 / Get the shape of wave dimensions

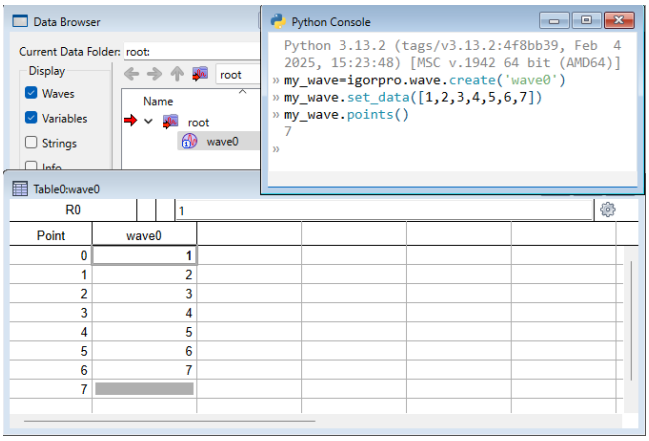


[Python]

```
my_wave=igorpro.wave.create('wave0',shape=(128,2))
my_wave.shape()
```

```
(128,2)
```

1D ウェーブのデータポイントの数を取得 / Get the number of data points for 1D wave

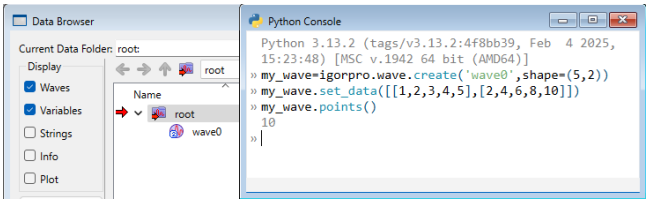


[Python]

```
my_wave=igorpro.wave.create('wave0')
my_wave.set_data([1,2,3,4,5,6,7])
my_wave.points()
```

```
7
```

2D ウェーブのデータポイントの数を取得 / Get the number of data points for 2D wave

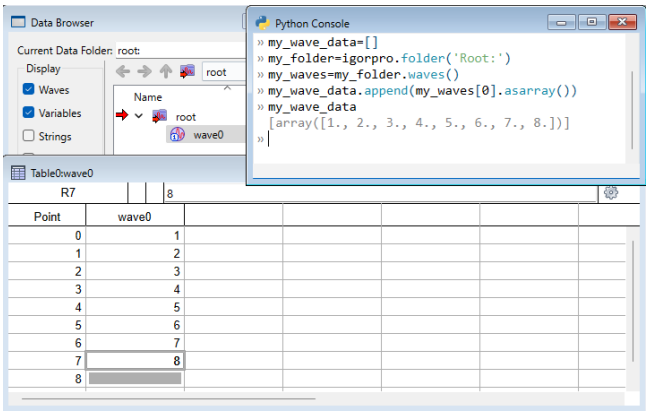


[Python]

```
my_wave=igorpro.wave.create('wave0',shape=(5,2))
my_wave.set_data([[1,2,3,4,5],[2,4,6,8,10]])
my_wave.points()
```

```
10
```

1D ウェーブデータを配列として取得 / Get a 1D wave data as an array



[Python]

```
my_wave_data=[]
my_wave_data.append(igorpro.wave('wave0').asarray())
```

リストの内容は次のようになります。/ The contents of the list are as follows.

[Python]

```
my_wave_data
```

```
[array([1., 2., 3., 4., 5.], dtype=float32)]
```

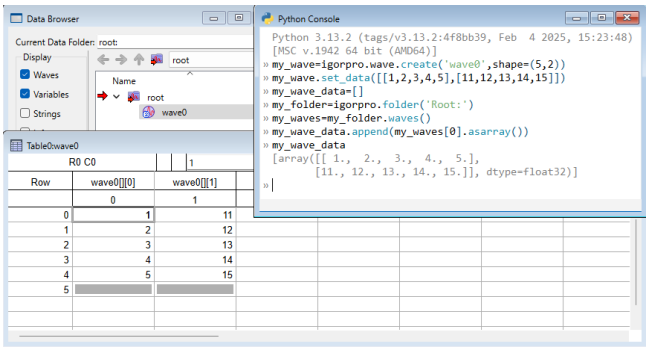
その後の処理の例（ウェーブの平均を計算）/ Example of subsequent processing (calculating the average of the wave)

[Python]

```
Import numpy as np
avg=np.average(my_wave_data)
avg
```

```
np.float32(3.0)
```

2D ウェーブデータを配列として取得 / Get a 2D wave data as an array



[Python]

```
my_wave_data=[]
my_wave_data.append(igorpro.wave('wave0').asarray())
```

リストの内容は次のようになります。/ The contents of the list are as follows.

[Python]

```
my_wave_data
```

```
[array([[ 1., 2., 3., 4., 5.],[11., 12., 13., 14., 15.]], dtype=float32)]
```

その後の処理の例（行列の転置）/ Example of subsequent processing (transposing a matrix)

[Python]

```
Import numpy as np
my_wave_data_2=[]
my_wave_data_2=np.transpose(my_wave_data)
my_wave_data_2
```

```
array([[[ 1.],[11.]],[[ 2.],[12.]],[[ 3.],[13.]],[[ 4.],[14.]],[[ 5.],[15.]]],
 dtype=float32)
```

変数の値の取得 / Get a variable value

Variable var1=2345

この var1 の値を取得する / Get the value of var1

[Python]

```
var=igorpro.variable('var1').value()
var
```

```
2345.0
```

画像オブジェクトを Python に渡す / Passing image objects to Python

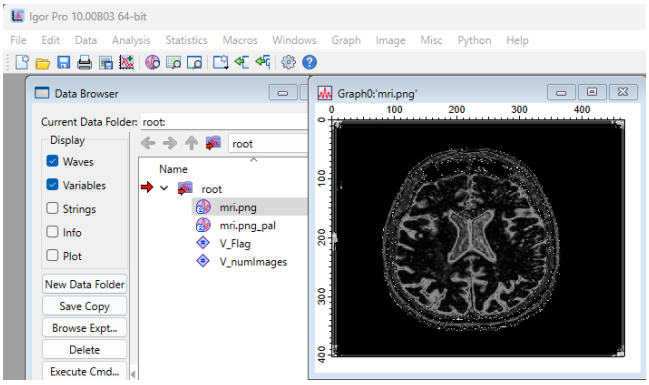
TensorFlow などの Python で使うことができる機械学習のパッケージで画像ファイルを処理させるには、Igor Pro にロードされている画像ファイルを渡す必要があります。ここでは、画像ファイルを Python に配列として渡す方法を解説します。また、渡されたデータが画像として扱えることを示すために、Pillow を使って Python 環境で PNG 画像を生成して表示します。/ To process image files using machine learning packages such as TensorFlow that can be used with Python, you need to pass the image files loaded in Igor Pro. Here, we will explain how to pass image files to Python as arrays. In addition, to demonstrate that the data provided can be treated as images, we will use Pillow to generate and display PNG images in Python environment.

Windows のコマンドプロンプトで Pillow をインストールしておく必要があります。/ You need to install Pillow in the Windows command prompt.

```
>pip install Pillow
```

1. グレースケール画像をロード / Loading a grayscale image

   画像ファイルは mri.png とします。/ Image file name is “mri.png”.

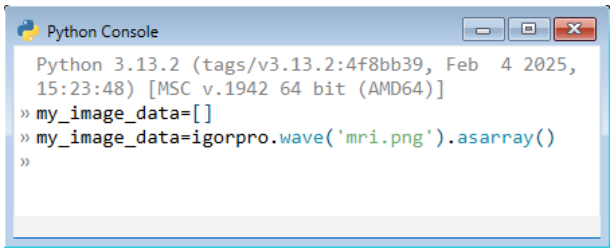


2. Python に画像を配列として読み込む / Reading the image as a Python array

[Python]

```
my_image_data=[]
my_image_data=igorpro.wave('mri.png').asarray()
```

my_image_data に mri.png が配列として読み込まれます。/ mri.png is loaded into my_image_data as an array.

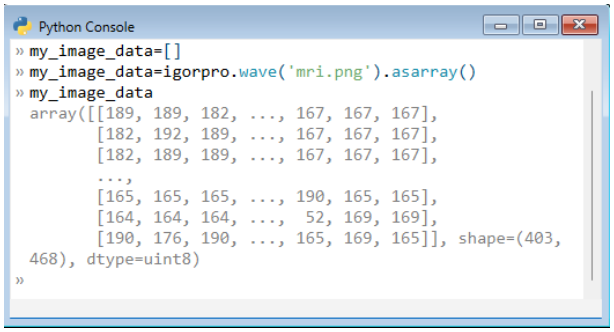


3. my_image_data の内容を確認してみます。/ Let’s check the content of my_image_data

[Python]

```
my_image_data
```

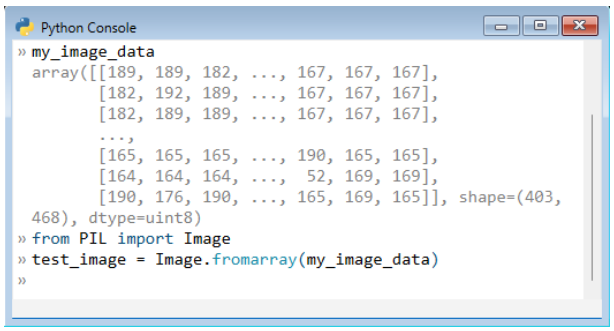
```
array([[189, 189, 182, ..., 167, 167, 167],
       [182, 192, 189, ..., 167, 167, 167],
       [182, 189, 189, ..., 167, 167, 167],
       ...,
       [165, 165, 165, ..., 190, 165, 165],
       [164, 164, 164, ...,  52, 169, 169],
       [190, 176, 190, ..., 165, 169, 165]],
      shape=(403, 468), dtype=uint8)
```



4. あとは、これを画像として処理するだけです。ここでは Pillow を使って配列を画像データに変換してみます。/ All that remains is to process this as an image. Here, we will use Pillow to convert the array into image data.

[Python]

```
from PIL import Image
test_image = Image.fromarray(my_image_data)
```

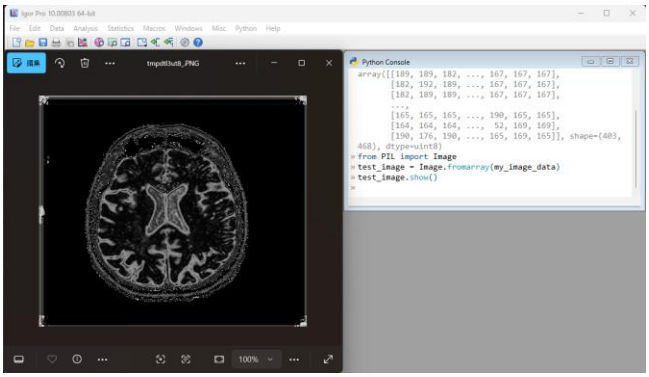


5. Pillow を使って、test_image に画像が入っていることを確認します。/ Use Pillow to confirm that the image is in test_image.

[Python]

```
test_image.show()
```

Windows のフォトが開いて、画像が表示されます。/ Windows Photos will open and the image will be displayed.



これで、この画像を `test_image.save('test_image.png')` でファイルとして保存したり、TensorFlow のモデルに入力画像として使うことができるようになります。/ Now you can save this image as a file using `test_image.save('test_image.png')` or use it as input image for the TensorFlow model.

TensourFlow を使った機械学習の例は、Running the MNIST Handwritten Digits Model with TensorFlow（machineLearning.ihf）を参照してください。/ For an example of machine learning using TensorFlow, see Running the MNIST Handwritten Digits Model with TensorFlow (machineLearning.ihf).

Igor Pro → Python / Igor Pro -> Python

Igor Pro のコマンドウィンドウから Python のコードを実行したり、.py ファイルを実行することができます。/ You can execute Python code or .py files from the Igor Pro command window.
構文は非常に簡単です。/ The syntax is very simple.

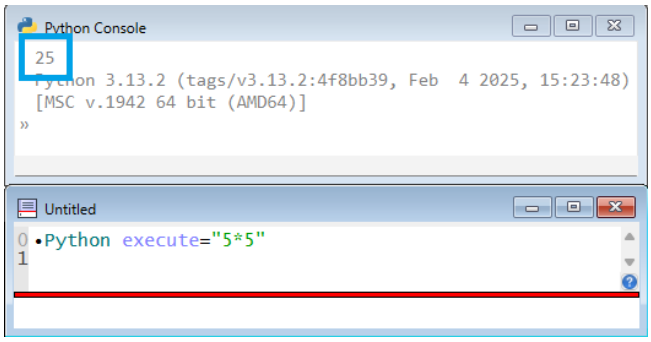
基本の構文は次のようになります：/ The basic syntax is as follows:

`Python execute = "Python のコード/Python code"`

例えば、コマンドウィンドウで次を実行すると、Python Console が開き、結果が表示されます。/ For example, if you execute the following in Command Window, the Python Console will open and display the results.

[Igor Pro]

```
Python execute="5*5"
```



変数の受け渡し / Passing variables

[Igor Pro]

```
Variable varA, varB
varA=12
varB=0
```

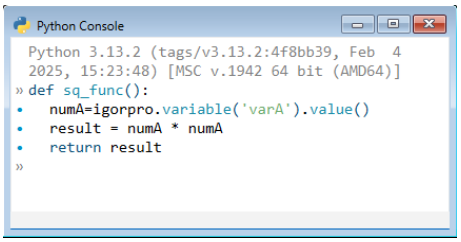
Python コードで varA を 2 乗して、varB に結果を保存する例を示します。/ Here is an example of squaring varA in Python code and storing the result in varB

```
Python execute = "a = sq_func()", var = {"a", varB}
```

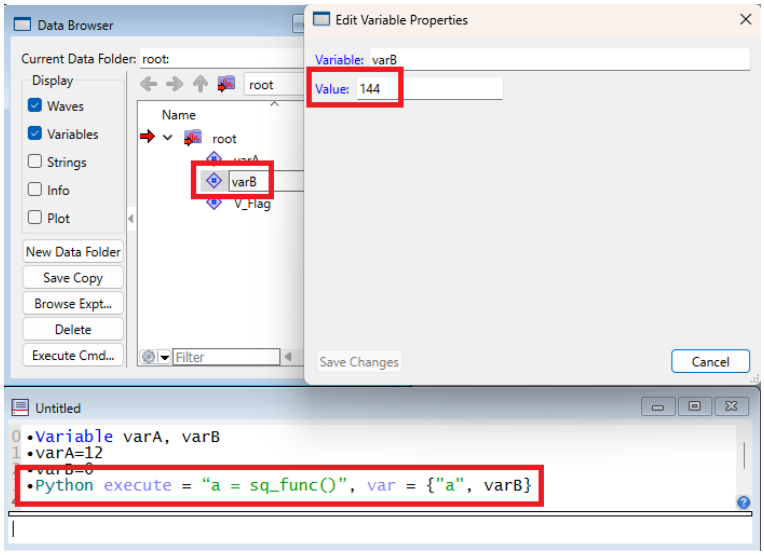
実行すると、計算結果が varB に代入されます。/ When executed, the calculation result is assigned to varB.

[Python]

```
def sq_func():
        numA=igorpro.variable('varA').value()
        result = numA * numA
        return result
```



Python が呼び出されると、`variable('varA').value()` で変数の値を読み込みます。/ When Python is called, the value of the variable is read using `variable('varA').value()`.



ウェーブの受け渡し / Passing waves

[Igor Pro]

```
Make/O/N=5 wave0, wave1
wave0={1,2,3,4,5}
```

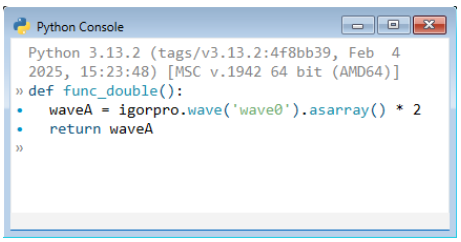
Python コードで wave0 の各要素に 2 を乗算して、wave1 に結果を保存する例を示します。/ Here is an example of multiplying each element of wave0 by 2 in Python code and storing the result in wave1.

```
Python execute = "a_wv = func_double()", array = {"a_wv", wave1}
```

実行すると、計算結果が wave1 に代入されます。/ When executed, the calculation result is assigned to wave1.

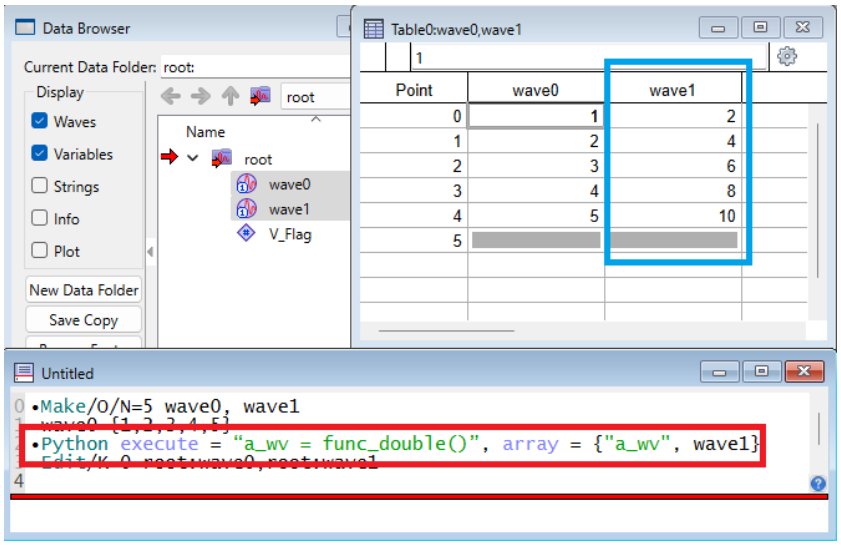
[Python]

```
def func_double():
        waveA =
            igorpro.wave('wave0').asarray()*2
        return waveA
```



Python が呼び出されると、`igorpro.wave('wave0').asarray()` でウェーブを配列の形式にしてを読み込みます。

/ When Python is called, the wave is read in array format using `igorpro.wave('wave0').asarray()`.



Python ファイル .py の呼び出し / Calling Python file .py

.py ファイルは Igor Pro が認識できる場所に配置しておく必要があります。Igor Pro フォルダーの User Procedure フォルダーに保存するのがわかりやすいです。/ .py files must be placed in locations that Igor Pro can recognize. It is easiest to save them in the User Procedure folder in the Igor Pro folder.

２種類の読み取り方法があるため、その例を示します。/ There are two different reading methods, so we will show you an example of each.

[Igor Pro]

```
Make/O/N=5 wave0, wave1
wave0={1,2,3,4,5}
```

Python の import 文の代わりに PythonFile コマンドを使う例を示します。/ Here is an example of using the PythonFile command instead of the Python “import” statement.

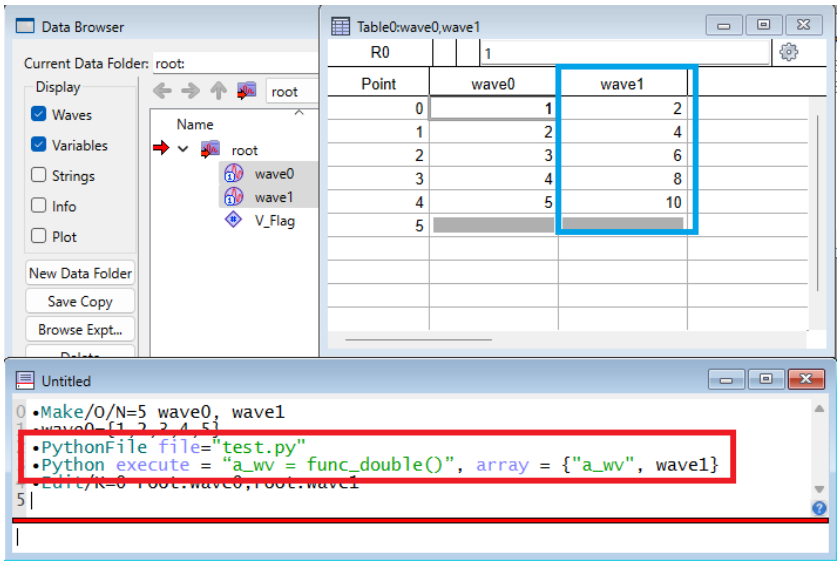
```
PythonFile file="test.py"
Python execute = “a_wv = func_double()”, array =
{"a_wv", wave1}
```

実行すると、計算結果が wave1 に代入されます。/ When executed, the calculation result is assigned to wave1.

[Python]

次のプログラムを”test.py”として、User Procedure フォルダーに保存します。

```
def func_double():
        waveA =
            igorpro.wave(‘wave0’).asarray()*2
        return waveA
```


Data Browser
Current Data Folder: root:
Display
Waves
Variables
Strings
Info
Plot
New Data Folder
Save Copy
Browse Expt...
root
Name
root
wave0
wave1
V_Flag
Table0:wave0,wave1
R0
1
Point
wave0
wave1
0 1 2
1 2 4
2 3 6
3 4 8
4 5 10
5
Untitled
0 •Make/O/N=5 wave0, wave1
•PythonFile file="test.py"
•Python execute = "a_wv = func_double()", array = {"a_wv", wave1}
5

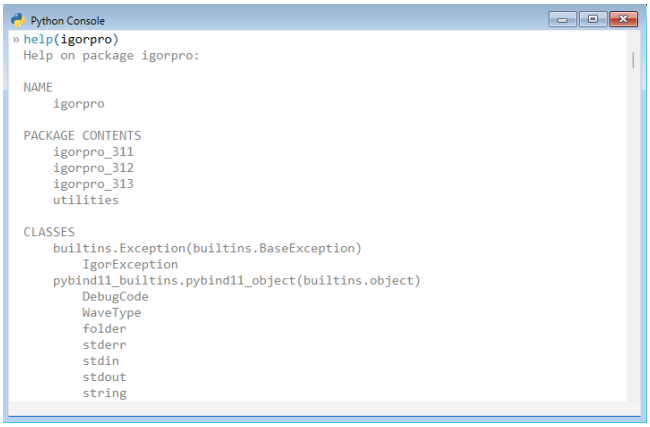
Appendix：Python で使うことができるオブジェクトの一覧を取得 / List of objects that can be used in Python

以下のコマンドで一覧を取得できます。/ You can get a list using the following commands.
igorpro.wave.create の各階層をしらべてみます。/ Let's take a look at each level of igorpro.wave.create.
まずは、igorpro オブジェクトそのものに関する情報です。/ First, here is some information about the 'igorpro' object.

[Python]

```
help(igorpro)
```

大量の情報が返ってきます。/ A large amount of information is returned.



igorpro の詳細 / Details of 'igorpro'

```
NAME
        igorpro

PACKAGE CONTENTS
        igorpro_311
        igorpro_312
        igorpro_313
        utilities

CLASSES
        builtins.Exception(builtins.BaseException)
        IgorException
        pybind11_builtins.pybind11_object(builtins.object)
        DebugCode
        WaveType
        folder
        stderr
        stdin
        stdout
        string
        variable
        wave
```

```
        class DebugCode(pybind11_builtins.pybind11_object)
        |       Method resolution order: / メソッドの解決の順序:
        |               DebugCode
        |               pybind11_builtins.pybind11_object
        |               builtins.object
        |
        |       Methods defined here: / メソッドの定義:
        |               __eq__(...)
        |                       __eq__(self: object, other: object) -> bool
        |               __getstate__(...)
        |                       __getstate__(self: object) -> int
        |               __hash__(...)
```

```
|                       __hash__(self: object) -> int
|               __index__(...)
|                       __index__(self: igorpro.DebugCode) -> int
|               __init__(...)
|                       __init__(self: igorpro.DebugCode, value: int) -> None
|               __int__(...)
|                       __int__(self: igorpro.DebugCode) -> int
|               __ne__(...)
|                       __ne__(self: object, other: object) -> bool
|               __repr__(...)
|                       __repr__(self: object) -> str
|               __setstate__(...)
|                       __setstate__(self: igorpro.DebugCode, state: int) -> None
|               __str__(...)
|                       __str__(self: object) -> str
|
| ----------------------------------------------------------------------
| Readonly properties defined here: / 読み取り専用のプロパティの定義:
|
| __members__
|
| name
|       name(self: object) -> str
|
| value
|
| ----------------------------------------------------------------------
| Data and other attributes defined here: / データとその他の属性の定義:
|
| AlreadyConnected = <DebugCode.AlreadyConnected: 2>
|
| FailedConnection = <DebugCode.FailedConnection: 1>
|
| SuccessfulConnection = <DebugCode.SuccessfulConnection: 0>
|
| ----------------------------------------------------------------------
| Static methods inherited from pybind11_builtins.pybind11_object:
| / pybind11_builtins.pybind11_object から継承した静的なメソッド:
|
| __new__(*args, **kwargs) class method of pybind11_builtins.pybind11_object
| Create and return a new object. See help(type) for accurate signature.
| / 新しいオブジェクトを作成して返す。正確なシグニチャは help(type) で確認。
```

```
class IgorException(builtins.Exception)
|       Method resolution order: / メソッドの解決の順序:
|               IgorException
|               builtins.Exception
|               builtins.BaseException
|               builtins.object
|
| Data descriptors defined here: / データの記述子の定義:
|
| __weakref__
|       list of weak references to the object / オブジェクトへの弱い参照のリスト。
|
| ----------------------------------------------------------------------
| Methods inherited from builtins.Exception: / builtins.Exception から継承したメソッド:
|
| __init__(self, /, *args, **kwargs)
| Initialize self. See help(type(self)) for accurate signature.
| / self を初期化。正確なシグニチャは help(type) で確認。
|
| ----------------------------------------------------------------------
| Static methods inherited from builtins.Exception:
| / builtins.Exception から継承した静的なメソッド:
|
| __new__(*args, **kwargs) class method of builtins.Exception
```

```
| Create and return a new object. See help(type) for accurate signature.
| / 新しいオブジェクトを作成して返す。正確なシグニチャは help(type) で確認。
|
| ----------------------------------------------------------------------
| Methods inherited from builtins.BaseException:
| / builtins.BaseException から継承したメソッド:
|
| __getattribute__(self, name, /)
|       Return getattr(self, name).
| __reduce__(self, /)
|       Helper for pickle.
| __repr__(self, /)
|       Return repr(self).
| __setstate__(self, object, /)
| __str__(self, /)
|       Return str(self).
| add_note(self, object, /)
|       Exception.add_note(note) --
|       add a note to the exception / 例外にノートを追加。
| with_traceback(self, object, /)
|       Exception.with_traceback(tb) --
|       set self.__traceback__ to tb and return self.
|       / tb に self.__traceback__ を設定し、self を返す。
|
| ----------------------------------------------------------------------
| Data descriptors inherited from builtins.BaseException:
| / builtins.BaseException から継承したデータ記述子。
|
| __cause__
|       exception cause
| __context__
|       exception context
| __dict__
| __suppress_context__
| __traceback__
| args
```

class WaveType(pybind11_builtins.pybind11_object)

```
| Method resolution order: / メソッドの解決の順序:
|       WaveType
|       pybind11_builtins.pybind11_object
|       builtins.object
|
| Methods defined here: / メソッドの定義:
|
| __eq__(...)
|       __eq__(self: object, other: object) -> bool
| __getstate__(...)
|       __getstate__(self: object) -> int
| __hash__(...)
|       __hash__(self: object) -> int
| __index__(...)
|       __index__(self: igorpro.WaveType) -> int
| __init__(...)
|       __init__(self: igorpro.WaveType, value: int) -> None
| __int__(...)
|       __int__(self: igorpro.WaveType) -> int
| __ne__(...)
|       __ne__(self: object, other: object) -> bool
| __repr__(...)
|       __repr__(self: object) -> str
| __setstate__(...)
|       __setstate__(self: igorpro.WaveType, state: int) -> None
| __str__(...)
|       __str__(self: object) -> str
|
| ----------------------------------------------------------------------
```

```
| Readonly properties defined here: / 読み取り専用のプロパティの定義:
|
| __members__
|
| name
|       name(self: object) -> str
|
| value
|
| ----------------------------------------------------------------------
| Data and other attributes defined here: / データとその他の属性の定義:
|
| complex128 = <WaveType.complex128: 5>
| complex64 = <WaveType.complex64: 3>
| dfref = <WaveType.dfref: 256>
| float32 = <WaveType.float32: 2>
| float64 = <WaveType.float64: 4>
| int16 = <WaveType.int16: 16>
| int32 = <WaveType.int32: 32>
| int64 = <WaveType.int64: 128>
| int8 = <WaveType.int8: 8>
| none = <WaveType.none: -1>
| text = <WaveType.text: 0>
| uint16 = <WaveType.uint16: 80>
| uint32 = <WaveType.uint32: 96>
| uint64 = <WaveType.uint64: 192>
| uint8 = <WaveType.uint8: 72>
| wref = <WaveType.wref: 16384>
|
| ----------------------------------------------------------------------
| Static methods inherited from pybind11_builtins.pybind11_object:
| / pybind11_builtins.pybind11_object から継承した静的なメソッド:
|
| __new__(*args, **kwargs) class method of pybind11_builtins.pybind11_object
| Create and return a new object. See help(type) for accurate signature.
| / 新しいオブジェクトを作成して返す。正確なシグニチャは help(type) で確認。
```

class folder(pybind11_builtins.pybind11_object)

```
| Represents an Igor data folder. / Igor データフォルダーに対応。
|
| Method resolution order: / メソッドの解決の順序:
|       folder
|       pybind11_builtins.pybind11_object
|       builtins.object
|
| Methods defined here: / メソッドの定義:
|
| __id__(...)
|       __id__(self: igorpro.folder) -> int
| __init__(...)
|       __init__(self: igorpro.folder, arg0: str) -> None
|       Constructs a Folder object from the indicated folder path.
|       / 指定されたフォルダーパスからフォルダーオブジェクトを構築。
| exists(...)
|       exists(self: igorpro.folder) -> bool
|       Returns the truth that the Igor data folder exists.
|       / Igor データフォルダーが存在するときは True を返す。
| kill(...)
|       kill(self: igorpro.folder, ignoreErrors: bool = False) -> None
|       Kills the Igor data folder. / Igor データフォルダーを Kill する。
| name(...)
|       name(self: igorpro.folder) -> str
|       Returns the name of the folder. / フォルダーの名前を返す。
| num_subfolders(...)
|       num_subfolders(self: igorpro.folder) -> int
|       Returns the number of subfolders in the parent folder.
|       / 親フォルダー内のサブフォルダーの数を返す。
```

```
| num_waves(...)
|       num_waves(self: igorpro.folder) -> int
|       Returns the number of waves in the folder. / フォルダー内のウェーブの数を返す。
| parent(...)
|       parent(self: igorpro.folder) -> igorpro.folder
|       Returns the parent data folder. / 親データフォルダーを返す。
| path(...)
|       path(self: igorpro.folder) -> str
|       Returns the full path to the folder. / フォルダーへのフルパスを返す。
| set(...)
|       set(self: igorpro.folder) -> None
|       Sets the folder as the current data folder. / フォルダーをカレントフォルダーに設定。
| subfolder(...)
|       subfolder(self: igorpro.folder, name: str) -> igorpro.folder
|       Returns the named subfolder as a PythonDataFolder.
|       / 指定されたサブフォルダーを PythonDataFolder として返す。
| subfolders(...)
|       subfolders(self: igorpro.folder) -> list
|       Returns a list of subfolders in the parent folder.
|       / 親フォルダー内のサブフォルダーのリストを返す。
| waves(...)
|       waves(self: igorpro.folder) -> list
|       Returns a python list of waves / ウェーブの Python リストを返す。
|
| ----------------------------------------------------------------------
| Static methods defined here: / 静的なメソッドの定義:
|
| create(...) method of builtins.PyCapsule instance
|       create(name: str) -> igorpro.folder
|       Creates a new data folder / 新しいデータフォルダーを作成。
| current(...) method of builtins.PyCapsule instance
|       current() -> igorpro.folder
|       Returns the current data folder as a PythonDataFolder.
|       / カレントデータフォルダーを PythonDataFolder として返す。
|
| ----------------------------------------------------------------------
| Static methods inherited from pybind11_builtins.pybind11_object:
| / pybind11_builtins.pybind11_object から継承した静的なメソッド:
|
| __new__(*args, **kwargs) class method of pybind11_builtins.pybind11_object
|       Create and return a new object. See help(type) for accurate signature.
|       / 新しいオブジェクトを作成して返す。正確なシグニチャは help(type) で確認。
```

```
class stderr(pybind11_builtins.pybind11_object)
| Standard error stream to Igor. / Igor への標準エラーストリーム。
|
| Method resolution order: / メソッドの解決の順序:
|       stderr
|       pybind11_builtins.pybind11_object
|       builtins.object
|
| Methods defined here: / メソッドの定義:
|
| __init__(...)
|       __init__(self: igorpro.stderr) -> None
|       Igor's standard error. / Igor の標準エラー。
| flush(...)
|       flush(self: igorpro.stderr) -> None
| write(...)
|       write(self: igorpro.stderr, arg0: str) -> None
|
| ----------------------------------------------------------------------
| Static methods inherited from pybind11_builtins.pybind11_object:
| / pybind11_builtins.pybind11_object から継承した静的なメソッド:
|
| __new__(*args, **kwargs) class method of pybind11_builtins.pybind11_object
```

```
|       Create and return a new object. See help(type) for accurate signature.
|       / 新しいオブジェクトを作成して返す。正確なシグニチャは help(type) で確認。
```

class stdin(pybind11_builtins.pybind11_object)

```
| Standard input stream to Igor. / Igor の標準入力ストリーム。
|
| Method resolution order: / メソッドの解決の順序:
|       stdin
|       pybind11_builtins.pybind11_object
|       builtins.object
|
| Methods defined here: / メソッドの定義:
|
| __init__(...)
|       __init__(self: igorpro.stdin) -> None
|       Igor's standard input. / Igor の標準入力。
|
| readline(...)
|       readline(self: igorpro.stdin, size: int = -1) -> str
|
| ----------------------------------------------------------------------
| Static methods inherited from pybind11_builtins.pybind11_object:
| / pybind11_builtins.pybind11_object から継承した静的なメソッド:
|
| __new__(*args, **kwargs) class method of pybind11_builtins.pybind11_object
|       Create and return a new object. See help(type) for accurate signature.
|       / 新しいオブジェクトを作成して返す。正確なシグニチャは help(type) で確認。
```

class stdout(pybind11_builtins.pybind11_object)

```
| Standard output stream to Igor. / Igor の標準出力ストリーム。
|
| Method resolution order: / メソッドの解決の順序:
|       stdout
|       pybind11_builtins.pybind11_object
|       builtins.object
|
| Methods defined here: / メソッドの定義:
|
| __init__(...)
|       __init__(self: igorpro.stdout) -> None
|       Igor's standard out. / Igor の標準出力。
| flush(...)
|       flush(self: igorpro.stdout) -> None
| write(...)
|       write(self: igorpro.stdout, arg0: str) -> None
|
| ----------------------------------------------------------------------
| Static methods inherited from pybind11_builtins.pybind11_object:
| / pybind11_builtins.pybind11_object から継承した静的なメソッド:
|
| __new__(*args, **kwargs) class method of pybind11_builtins.pybind11_object
|       Create and return a new object. See help(type) for accurate signature.
|       / 新しいオブジェクトを作成して返す。正確なシグニチャは help(type) で確認。
```

class string(pybind11_builtins.pybind11_object)

```
| Represents an Igor string variable. / Igor の文字列変数に対応。
|
| Method resolution order: / メソッドの解決の順序:
|       string
|       pybind11_builtins.pybind11_object
|       builtins.object
|
| Methods defined here: / メソッドの定義:
|
| __init__(...)
```

```
|       __init__(self: igorpro.string, name: str, folder: igorpro.folder =
<igorpro.folder object at 0x000001A3F2DAF5B0>) -> None
|       Constructs a PythonString object from the indicated path.
|       / 指定されたパスから PythonString オブジェクトを構築。
| exists(...)
|       exists(self: igorpro.string) -> bool
|       Returns the truth that the Igor string variable exists.
|       / Igor 文字列変数が存在しているかどうかを返す。
| kill(...)
|       kill(self: igorpro.string, ignoreErrors: bool = False) -> None
|       Kills the Igor string variable. / Igor 文字列変数を Kill する。
| name(...)
|       name(self: igorpro.string) -> str
|       Returns the name of the Igor string. / Igor 文字列の名前を返す。
| path(...)
|       path(self: igorpro.string) -> str
|       Returns the full path to the Igor string, not including its name.
|       / Igor 文字列へのフルパスを、その名前を含まずに返す。
| set(...)
|       set(self: igorpro.string, str: str) -> None
|       Sets the value of the Igor string. / Igor 文字列の値を設定する。
| value(...)
|       value(self: igorpro.string) -> str
|       Returns the value of the Igor string. / Igor 文字列の値を返す。
|
| ----------------------------------------------------------------------
| Static methods defined here: / 静的なメソッドの定義:
|
| create(...) method of builtins.PyCapsule instance
|       create(*args, **kwargs)
|       Overloaded function.
|
|       1. create(name: str, value: str = '', overwrite: bool = False) -> igorpro.string
|       Creates a new Igor string relative to the current data folder.
|       / 現在のデータフォルダーに対して新しい Igor 文字列を作成する。
|
|       2. create(name: str, folder: igorpro.folder, value: str = '', overwrite: bool =
False) -> igorpro.string
|       Creates a new Igor string relative to the specified data folder.
|       / 指定されたデータフォルダーに関連する新しい Igor 文字列を作成する。
|
| ----------------------------------------------------------------------
| Static methods inherited from pybind11_builtins.pybind11_object:
| / pybind11_builtins.pybind11_object から継承した静的なメソッド:
|
| __new__(*args, **kwargs) class method of pybind11_builtins.pybind11_object
|       Create and return a new object. See help(type) for accurate signature.
|       / 新しいオブジェクトを作成して返す。正確なシグニチャは help(type) で確認。
```

class variable(pybind11_builtins.pybind11_object)

```
| Represents a numeric variable in Igor Pro. / Igor Pro の数値変数に対応。
|
| Method resolution order: / メソッドの解決の順序:
|       variable
|       pybind11_builtins.pybind11_object
|       builtins.object
|
| Methods defined here: / メソッドの定義:
|
| __init__(...)
|       __init__(self: igorpro.variable, name: str, folder: igorpro.folder =
<igorpro.folder object at 0x000001A3F2DAE470>) -> None
|       Constructs a PythonVariable object from an existing Igor numeric variable.
|       / 既存の Igor 数値変数から PythonVariable オブジェクトを構築する。
| exists(...)
|       exists(self: igorpro.variable) -> bool
|       Returns the truth that the Igor variable exists.
```

```
|       / Igor 変数が存在すれば真を返す。
| imag(...)
|       imag(self: igorpro.variable) -> object
|       Returns the imaginary part of the Igor variable. Non-complex types return 0.
|       / Igor 変数の虚数部を返す。複素数型でない場合は 0 を返す。
| iscomplex(...)
|       iscomplex(self: igorpro.variable) -> bool
|       Returns truth that the variable is a complex type.
|       / 変数が複素数型の場合は真を返す。
| kill(...)
|       kill(self: igorpro.variable, ignoreErrors: bool = False) -> None
|       Kills the Igor variable. / Igor 変数を Kill する。
| name(...)
|       name(self: igorpro.variable) -> str
|       Returns the name of the variable. / 変数の名前を返す。
| path(...)
|       path(self: igorpro.variable) -> str
|       Returns the full Igor path to the variable. / 変数へのフルパスを返す。
| real(...)
|       real(self: igorpro.variable) -> object
|       Returns the real part of the Igor variable. / Igor 変数の実部を返す。
| set(...)
|       set(self: igorpro.variable, value: object) -> None
|       Sets the value of the variable. / 変数の値を設定する。
| value(...)
|       value(self: igorpro.variable) -> object
|       Returns the value of the variable. / 変数の値を返す。
|
| ----------------------------------------------------------------------
| Static methods defined here: / 静的なメソッドの定義:
|
| create(...) method of builtins.PyCapsule instance
|       create(*args, **kwargs)
|       Overloaded function.
|
|       1. create(name: str, value: object = 0, overwrite: bool = False) ->
igorpro.variable
|       Creates a new Python variable with the specified value.
|       / 指定した値を持つ新しい Python 変数を作成する。
|
|       2. create(name: str, folder: igorpro.folder, value: object = 0, overwrite: bool =
False) -> igorpro.variable
|       Creates a new Python variable with the specified value.
|       / 指定した値を持つ新しい Python 変数を作成する。
|
| ----------------------------------------------------------------------
| Static methods inherited from pybind11_builtins.pybind11_object:
| / pybind11_builtins.pybind11_object から継承した静的なメソッド:
|
| __new__(*args, **kwargs) class method of pybind11_builtins.pybind11_object
|       Create and return a new object. See help(type) for accurate signature.
|       / 新しいオブジェクトを作成して返す。正確なシグニチャは help(type) で確認。
```

class wave(pybind11_builtins.pybind11_object)

```
| Represents an Igor wave. / Igor のウェーブに対応。
|
| Method resolution order: / メソッドの解決の順序:
|       wave
|       pybind11_builtins.pybind11_object
|       builtins.object
|
| Methods defined here: / メソッドの定義:
|
| __getitem__(...)
|       __getitem__(*args, **kwargs)
|       Overloaded function.
|       1. __getitem__(self: igorpro.wave, arg0: int) -> object
|       2. __getitem__(self: igorpro.wave, arg0: tuple) -> object
```

```
| __id__(...)
|       __id__(self: igorpro.wave) -> int
| __init__(...)
|       __init__(self: igorpro.wave, name: str, folder: igorpro.folder = <igorpro.folder
object at 0x000001A3F2DAB2F0>) -> None
|       Constructs a Wave object from the indicated wave path.
|       / 指定されたウェーブパスからウェーブオブジェクトを構築する。
| __setitem__(...)
|       __setitem__(*args, **kwargs)
|       Overloaded function.
|       1. __setitem__(self: igorpro.wave, arg0: int, arg1: object) -> None
|       2. __setitem__(self: igorpro.wave, arg0: tuple, arg1: object) -> None
| asarray(...)
|       asarray(self: igorpro.wave) -> numpy.ndarray
|       Converts an Igor wave to a numpy array / Igor ウェーブを numpy 配列に変換する。
| dims(...)
|       dims(self: igorpro.wave) -> int
|       Returns the number of dimensions in the wave. / ウェーブ内の次元数を返す。
| exists(...)
|       exists(self: igorpro.wave) -> bool
|       Returns truth that the wave exists. / ウェーブが存在するときは真を返す。
| kill(...)
|       kill(self: igorpro.wave, ignoreErrors: bool = False) -> None
|       Kills the wave if it isn't in use. / ウェーブが使われていないときはそれを Kill する。
| labels(...)
|       labels(*args, **kwargs)
|       Overloaded function.
|       1. labels(self: igorpro.wave, dimension: str) -> list
|       Returns the wave's dimension labels for the specified dimension as a list
|       / 指定された次元のウェーブの次元ラベルをリストとして返す。
|       2. labels(self: igorpro.wave, dimension: int) -> list
|       Returns the wave's dimension labels for the specified dimension as a list
|       / 指定された次元のウェーブの次元ラベルをリストとして返す。
| name(...)
|       name(self: igorpro.wave) -> str
|       Returns the name of the wave. / ウェーブの名前を返す。
| note(...)
|       note(self: igorpro.wave) -> str
|       Returns the wave's note string. / ウェーブのノート文字列を返す。
| path(...)
|       path(self: igorpro.wave) -> str
|       Returns the full path to the wave, not including its name.
|       / ウェーブへのフルパスを、名前を含まずに返す。
| points(...)
|       points(self: igorpro.wave) -> int
|       Returns the number of data points in the wave.
|       / ウェーブ内のデータポイントの数を返す。
| redimension(...)
|       redimension(*args, **kwargs)
|       Overloaded function.
|       1. redimension(self: igorpro.wave, shape: int) -> None
|       Redimensions the wave to the indicated dimensions.
|       / 指示された次元にウェーブをリサイズする。
|       2. redimension(self: igorpro.wave, shape: tuple) -> None
|       Redimensions the wave to the indicated dimensions.
|       / 指示された次元にウェーブをリサイズする。
| scale(...)
|       scale(*args, **kwargs)
|       Overloaded function.
|       1. scale(self: igorpro.wave, dimension: str) -> tuple
|       Returns the wave's delta and offset for the specified dimension
|       / 指定された次元のウェーブのデルタとオフセットを返す。
|       2. scale(self: igorpro.wave, dimension: int) -> tuple
|       Returns the wave's delta and offset for the specified dimension
|       / 指定された次元のウェーブのデルタとオフセットを返す。
| set_data(...)
|       set_data(*args, **kwargs)
|       Overloaded function.
```

```
|       1. set_data(self: igorpro.wave, arg0: igorpro.wave) -> None
|       Sets the wave's data, and all of it's meta-data, to the input wave
|       / 入力ウェーブにウェーブのデータとメタデータをすべて設定する。
|       2. set_data(self: igorpro.wave, arg0: object) -> None
|       Sets the wave's data to the input array
|       / 入力ウェーブにウェーブのデータを設定する。
| set_label(...)
|       set_label(*args, **kwargs)
|       Overloaded function.
|       set_label(self: igorpro.wave, dimension: str, index: int, label: str) -> None
|       Sets the wave's dimension labels for the specified dimension as a list
|       / 指定された次元のウェーブの次元ラベルをリストとして設定する。
| set_scale(...)
|       set_scale(*args, **kwargs)
|       Overloaded function.
|       1. set_scale(self: igorpro.wave, dimension: str, start: float, value: float,
mode: str) -> None
|       Sets the scaling of the wave for the indicated dimension, given start, value, and
mode. / 指定された次元のウェーブのスケーリングを設定する。開始、値、モードを指定する。
|       2. set_scale(self: igorpro.wave, dimension: int, start: float, value: float,
mode: str) -> None
|       Sets the scaling of the wave for the indicated dimension, given start, value, and
mode.  / 指定された次元のウェーブのスケーリングを設定する。開始、値、モードを指定する。
| set_units(...)
|       set_units(*args, **kwargs)
|       Overloaded function.
|       1. set_units(self: igorpro.wave, dimension: str, unit: str) -> None
|       Sets the wave's unit string for the specified dimension
|       / 指定された次元のウェーブの単位文字列を設定する。
| shape(...)
|       shape(self: igorpro.wave) -> tuple
|       Returns the shape of the wave dimensions as a tuple.
|       / タプルとしてウェーブの次元の形状を返す。
| stats(...)
|       stats(self: igorpro.wave, xrange: object = [Ellipsis, Ellipsis], yrange: object =
[Ellipsis, Ellipsis], zrange: object = [Ellipsis, Ellipsis], trange: object = [Ellipsis,
Ellipsis], method: str = 'real') -> dict
|       Calculates the wave statistics with an optional multi-dimensional range.
|       / オプションの多次元範囲でウェーブの統計を計算する。
| type(...)
|       type(self: igorpro.wave) -> igorpro.WaveType
|       Returns the data type of the wave. / ウェーブのデータ形式を返す。
| units(...)
|       units(*args, **kwargs)
|       Overloaded function.
|       1. units(self: igorpro.wave, dimension: str) -> str
|       Returns the wave's unit string for the specified dimension
|       / 指定された次元のウェーブの単位文字列を返す。
|       2. units(self: igorpro.wave, dimension: int) -> str
|       Returns the wave's unit string for the specified dimension
|       / 指定された次元のウェーブの単位文字列を返す。
|
| ----------------------------------------------------------------------
| Static methods defined here: / 静的なメソッドの定義:
|
| create(...) method of builtins.PyCapsule instance
|       create(*args, **kwargs)
|       Overloaded function.
|
|       1. create(name: str, shape: int = 128, value: object = None, type:
igorpro.WaveType = <WaveType.none: -1>, overwrite: bool = False) -> igorpro.wave
|       Creates a new Igor wave, and returns a PythonWave reference to it.
|       / 新しい Igor ウェーブを作成し、PythonWave の参照を返す。
|
|       2. create(name: str, folder: igorpro.folder, shape: int = 128, value: object =
None, type: igorpro.WaveType = <WaveType.none: -1>, overwrite: bool = False) ->
igorpro.wave
|       Creates a new Igor wave, and returns a PythonWave reference to it.
```

```
|       / 新しい Igor ウェーブを作成し、PythonWave の参照を返す。
|
|       3. create(name: str, shape: tuple = (128,), value: object = None, type:
igorpro.WaveType = <WaveType.none: -1>, overwrite: bool = False) -> igorpro.wave
|       Creates a new Igor wave, and returns a PythonWave reference to it.
|       / 新しい Igor ウェーブを作成し、PythonWave の参照を返す。
|
|       4. create(name: str, folder: igorpro.folder, shape: tuple = (128,), value: object
= None, type: igorpro.WaveType = <WaveType.none: -1>, overwrite: bool = False) ->
igorpro.wave
|       Creates a new Igor wave, and returns a PythonWave reference to it.
|       / 新しい Igor ウェーブを作成し、PythonWave の参照を返す。
|
| createfrom(...) method of builtins.PyCapsule instance
|       createfrom(*args, **kwargs)
|       Overloaded function.
|
|       1. createfrom(name: str, source: object, overwrite: bool = False) -> igorpro.wave
|       Creates a new Igor wave, using an array-like object as the source data. Returns a
PythonWave reference to it.
|       / 配列のようなオブジェクトをソースデータとして使用して、新しい Igor ウェーブを作る。PythonWave
への参照を返す。
|
|       2. createfrom(name: str, folder: igorpro.folder, source: object, overwrite: bool
= False) -> igorpro.wave
|       Creates a new Igor wave, using an array-like object as the source data. Returns a
PythonWave reference to it.
|       / 配列のようなオブジェクトをソースデータとして使用して、新しい Igor ウェーブを作る。PythonWave
への参照を返す。
|
| ----------------------------------------------------------------------
| Static methods inherited from pybind11_builtins.pybind11_object:
| / pybind11_builtins.pybind11_object から継承した静的なメソッド:
|
| __new__(*args, **kwargs) class method of pybind11_builtins.pybind11_object
|       Create and return a new object. See help(type) for accurate signature.
|       / 新しいオブジェクトを作成して返す。正確なシグニチャは help(type)で確認。


FUNCTIONS
        bombout(...) method of builtins.PyCapsule instance
                bombout() -> str
        build(...) method of builtins.PyCapsule instance
                build() -> str
        connect(...) method of builtins.PyCapsule instance
                connect() -> PythonDebugCode
        execute(...) method of builtins.PyCapsule instance
                execute(code: str, ignoreErrors: bool = False) -> None
        ignore(...) method of builtins.PyCapsule instance
                ignore(arg0: object) -> None
        print(...) method of builtins.PyCapsule instance
                print(arg0: str) -> None
        version(...) method of builtins.PyCapsule instance
                version() -> str

DATA
        complex128 = <WaveType.complex128: 5>
        complex64 = <WaveType.complex64: 3>
        dfref = <WaveType.dfref: 256>
        float32 = <WaveType.float32: 2>
        float64 = <WaveType.float64: 4>
        int16 = <WaveType.int16: 16>
        int32 = <WaveType.int32: 32>
        int64 = <WaveType.int64: 128>
        int8 = <WaveType.int8: 8>
        text = <WaveType.text: 0>
        uint16 = <WaveType.uint16: 80>
        uint32 = <WaveType.uint32: 96>
```

```
        uint64 = <WaveType.uint64: 192>
        uint8 = <WaveType.uint8: 72>
        wref = <WaveType.wref: 16384>

FILE
        c:¥program files¥wavemetrics¥igor pro 10
        folder¥igorbinaries_x64¥python¥igorpro¥__init__.py
```

wave の詳細 / Details of 'wave'

Igorpro オブジェクトに含まれる wave オブジェクトに関する情報を見てみます。/ Let's take a look at the information about 'wave' object contained in 'igorpro' object.

```
help(igorpro.wave)
```

```
class wave(pybind11_builtins.pybind11_object)
| Represents an Igor wave. / Igor のウェーブに対応。
|
| Method resolution order: / メソッドの解決の順序:
|       wave
|       pybind11_builtins.pybind11_object
|       builtins.object
|
| Methods defined here: / メソッドの定義:
|
| __getitem__(...)
|       __getitem__(*args, **kwargs)
|       Overloaded function.
|       1. __getitem__(self: igorpro.wave, arg0: int) -> object
|       2. __getitem__(self: igorpro.wave, arg0: tuple) -> object
|
| __id__(...)
|       __id__(self: igorpro.wave) -> int
|
| __init__(...)
|       __init__(self: igorpro.wave, name: str, folder: igorpro.folder = <igorpro.folder object
at 0x000001A3F2DAB2F0>) -> None
|       Constructs a Wave object from the indicated wave path.
|       / 指定されたウェーブパスからウェーブオブジェクトを構築する。
| __setitem__(...)
|       __setitem__(*args, **kwargs)
|       Overloaded function.
|       1. __setitem__(self: igorpro.wave, arg0: int, arg1: object) -> None
|       2. __setitem__(self: igorpro.wave, arg0: tuple, arg1: object) -> None
| asarray(...)
|       asarray(self: igorpro.wave) -> numpy.ndarray
|       Converts an Igor wave to a numpy array / Igor ウェーブを numpy 配列に変換する。
| dims(...)
|       dims(self: igorpro.wave) -> int
|       Returns the number of dimensions in the wave. / ウェーブの次元数を返す。
| exists(...)
|       exists(self: igorpro.wave) -> bool
|       Returns truth that the wave exists. / ウェーブが存在するときは真を返す。
| kill(...)
|       kill(self: igorpro.wave, ignoreErrors: bool = False) -> None
|       Kills the wave if it isn't in use. / ウェーブが使われていないときは Kill する。
| labels(...)
|       labels(*args, **kwargs)
|       Overloaded function.
|       1. labels(self: igorpro.wave, dimension: str) -> list
|       Returns the wave's dimension labels for the specified dimension as a list
```

```
|       / 指定された次元のウェーブの次元ラベルをリストとして返す。
|       2. labels(self: igorpro.wave, dimension: int) -> list
|       Returns the wave's dimension labels for the specified dimension as a list
|       / 指定された次元のウェーブの次元ラベルをリストとして返す。
| name(...)
|       name(self: igorpro.wave) -> str
|       Returns the name of the wave. / ウェーブの名前を返す。
| note(...)
|       note(self: igorpro.wave) -> str
|       Returns the wave's note string. / ウェーブのノートの文字列を返す。
| path(...)
|       path(self: igorpro.wave) -> str
|       Returns the full path to the wave, not including its name.
|       / ウェーブへのフルパスを、名前を含まずに返す。
| points(...)
|       points(self: igorpro.wave) -> int
|       Returns the number of data points in the wave. / ウェーブ内のデータポイントの数を返す。
| redimension(...)
|       redimension(*args, **kwargs)
|       Overloaded function.
|       1. redimension(self: igorpro.wave, shape: int) -> None
|       Redimensions the wave to the indicated dimensions.
|       / ウェーブを指定された次元に変更する。
|       2. redimension(self: igorpro.wave, shape: tuple) -> None
|       Redimensions the wave to the indicated dimensions.
|       / ウェーブを指定された次元に変更する。
| scale(...)
|       scale(*args, **kwargs)
|       Overloaded function.
|       1. scale(self: igorpro.wave, dimension: str) -> tuple
|       Returns the wave's delta and offset for the specified dimension
|       / 指定された次元のウェーブのデルタとオフセットを返す。
|       2. scale(self: igorpro.wave, dimension: int) -> tuple
|       Returns the wave's delta and offset for the specified dimension
|       / 指定された次元のウェーブのデルタとオフセットを返す。
| set_data(...)
|       set_data(*args, **kwargs)
|       Overloaded function.
|       1. set_data(self: igorpro.wave, arg0: igorpro.wave) -> None
|       Sets the wave's data, and all of it's meta-data, to the input wave
|       / 入力ウェーブにウェーブのデータとメタデータをすべて設定する。
|       2. set_data(self: igorpro.wave, arg0: object) -> None
|       Sets the wave's data to the input array / ウェーブのデータを入力配列に設定する。
| set_label(...)
|       set_label(*args, **kwargs)
|       Overloaded function.
|       1. set_label(self: igorpro.wave, dimension: str, index: int, label: str) -> None
|       Returns the wave's dimension labels for the specified dimension as a list
|       / 指定された次元のウェーブの次元ラベルをリストとして返す。
|       2. set_label(self: igorpro.wave, dimension: int, index: int, label: str) -> None
|       Returns the wave's dimension labels for the specified dimension as a list
|       / 指定された次元のウェーブの次元ラベルをリストとして返す。
| set_scale(...)
|       set_scale(*args, **kwargs)
|       Overloaded function.
|       1. set_scale(self: igorpro.wave, dimension: str, start: float, value: float, mode: str) -
> None
|       Sets the scaling of the wave for the indicated dimension, given start, value, and mode.
|       / 指定された次元のウェーブのスケーリングを設定する。開始、値、モードを指定する。
|       2. set_scale(self: igorpro.wave, dimension: int, start: float, value: float, mode: str) -
> None
|       Sets the scaling of the wave for the indicated dimension, given start, value, and mode.
|       / 指定された次元のウェーブのスケーリングを設定する。開始、値、モードを指定する。
| set_units(...)
|       set_units(*args, **kwargs)
|       Overloaded function.
|       1. set_units(self: igorpro.wave, dimension: str, unit: str) -> None
|       Sets the wave's unit string for the specified dimension
```

```
|       / 指定された次元のウェーブの単位文字列を返す。
|       2. set_units(self: igorpro.wave, dimension: int, unit: str) -> None
|       Sets the wave's unit string for the specified dimension
|       / 指定された次元のウェーブの単位文字列を返す。
| shape(...)
|       shape(self: igorpro.wave) -> tuple
|       Returns the shape of the wave dimensions as a  tuple. / タプルとしてウェーブの次元の形状を返す。
| stats(...)
|       stats(self: igorpro.wave, xrange: object = [Ellipsis, Ellipsis], yrange: object =
[Ellipsis, Ellipsis], zrange: object = [Ellipsis, Ellipsis], trange: object = [Ellipsis,
Ellipsis], method: str = 'real') -> dict
|       Calculates the wave statistics with an optional multi-dimensional range.
|       / オプションの多次元範囲でウェーブ統計を計算する。
| type(...)
|       type(self: igorpro.wave) -> igorpro.WaveType
|       Returns the data type of the wave. / ウェーブのデータ形式を返す。
| units(...)
|       units(*args, **kwargs)
|       Overloaded function.
|       1. units(self: igorpro.wave, dimension: str) -> str
|       Returns the wave's unit string for the specified dimension
|       / 指定された次元のウェーブの単位文字列を返す。
|       2. units(self: igorpro.wave, dimension: int) -> str
|       Returns the wave's unit string for the specified dimension
|       / 指定された次元のウェーブの単位文字列を返す。
|
| ----------------------------------------------------------------------
| Static methods defined here: / 静的なメソッドの定義:
|
| create(...) method of builtins.PyCapsule instance
|       create(*args, **kwargs)
|       Overloaded function.
|       1. create(name: str, shape: int = 128, value: object = None, type: igorpro.WaveType =
<WaveType.none: -1>, overwrite: bool = False) -> igorpro.wave
|       Creates a new Igor wave, and returns a PythonWave reference to it.
|       / 新しい Igor ウェーブを作成し、PythonWave の参照を返す。
|       2. create(name: str, folder: igorpro.folder, shape: int = 128, value: object = None,
type: igorpro.WaveType = <WaveType.none: -1>, overwrite: bool = False) -> igorpro.wave
|       Creates a new Igor wave, and returns a PythonWave reference to it.
|       / 新しい Igor ウェーブを作成し、PythonWave の参照を返す。
|       3. create(name: str, shape: tuple = (128,), value: object = None, type: igorpro.WaveType
= <WaveType.none: -1>, overwrite: bool = False) -> igorpro.wave
|       Creates a new Igor wave, and returns a PythonWave reference to it.
|       / 新しい Igor ウェーブを作成し、PythonWave の参照を返す。
|       4. create(name: str, folder: igorpro.folder, shape: tuple = (128,), value: object = None,
type: igorpro.WaveType = <WaveType.none: -1>, overwrite: bool = False) -> igorpro.wave
|       Creates a new Igor wave, and returns a PythonWave reference to it.
|       / 新しい Igor ウェーブを作成し、PythonWave の参照を返す。
| createfrom(...) method of builtins.PyCapsule instance
|       createfrom(*args, **kwargs)
|       Overloaded function.
|       1. createfrom(name: str, source: object, overwrite: bool = False) -> igorpro.wave
|       Creates a new Igor wave, using an array-like object as the source data. Returns a
PythonWave reference to it.
|       / 配列のようなオブジェクトをソースデータとして使用して、新しい Igor ウェーブを作成する。PythonWave への
参照を返す。
|       2. createfrom(name: str, folder: igorpro.folder, source: object, overwrite: bool = False)
-> igorpro.wave
|       Creates a new Igor wave, using an array-like object as the source data. Returns a
PythonWave reference to it.
|       / 配列のようなオブジェクトをソースデータとして使用して、新しい Igor ウェーブを作成する。PythonWave への
参照を返す。
|
| ----------------------------------------------------------------------
| Static methods inherited from pybind11_builtins.pybind11_object:
| / pybind11_builtins.pybind11_object から継承した静的なメソッド:
|
| __new__(*args, **kwargs) class method of pybind11_builtins.pybind11_object
```

```
|       Create and return a new object. See help(type) for accurate signature.
|       / 新しいオブジェクトを作成して返す。正確なシグニチャは help(type) で確認。
```

folder の詳細 / Details of 'folder'

```
help(igorpro.folder)
```

```
class folder(pybind11_builtins.pybind11_object)
| Represents an Igor data folder. / Igor データフォルダーに対応。
|
| Method resolution order: / メソッドの解決の順序:
|       folder
|       pybind11_builtins.pybind11_object
|       builtins.object
|
| Methods defined here: / メソッドの定義:
|
| __id__(...)
|       __id__(self: igorpro.folder) -> int
|
| __init__(...)
|       __init__(self: igorpro.folder, arg0: str) -> None
|       Constructs a Folder object from the indicated folder path.
|       / 指定されたフォルダーパスからフォルダーオブジェクトを構築する。
| exists(...)
|       exists(self: igorpro.folder) -> bool
|       Returns the truth that the Igor data folder exists.
|       / Igor データフォルダーが存在する時に真を返す。
| kill(...)
|       kill(self: igorpro.folder, ignoreErrors: bool = False) -> None
|       Kills the Igor data folder. / Igor データフォルダーを Kill する。
| name(...)
|       name(self: igorpro.folder) -> str
|       Returns the name of the folder. / フォルダーの名前を返す。
| num_subfolders(...)
|       num_subfolders(self: igorpro.folder) -> int
|       Returns the number of subfolders in the parent folder.
|       / 親フォルダー内のサブフォルダーの数を返す。
| num_waves(...)
|       num_waves(self: igorpro.folder) -> int
|       Returns the number of waves in the folder. / フォルダー内のウェーブの数を返す。
| parent(...)
|       parent(self: igorpro.folder) -> igorpro.folder
|       Returns the parent data folder. / 親データフォルダーを返す。
| path(...)
|       path(self: igorpro.folder) -> str
|       Returns the full path to the folder. / フォルダーへのフルパスを返す。
| set(...)
|       set(self: igorpro.folder) -> None
|       Sets the folder as the current data folder. / フォルダーをカレントデータフォルダーにする。
| subfolder(...)
|       subfolder(self: igorpro.folder, name: str) -> igorpro.folder
|       Returns the named subfolder as a PythonDataFolder.
|       / 指定されたサブフォルダーを PythonDataFolder として返す。
| subfolders(...)
|       subfolders(self: igorpro.folder) -> list
|       Returns a list of subfolders in the parent folder.
|       / 親フォルダー内のサブフォルダーの一覧を返す。
| waves(...)
|       waves(self: igorpro.folder) -> list
|       Returns a python list of waves / ウェーブの Python リストを返す。
```

```
|
| ----------------------------------------------------------------------
| Static methods defined here: / 静的なメソッドの定義:
|
| create(...) method of builtins.PyCapsule instance
|       create(name: str) -> igorpro.folder
|       Creates a new data folder / 新しいデータフォルダーを作成する。
| current(...) method of builtins.PyCapsule instance
|       current() -> igorpro.folder
|       Returns the current data folder as a PythonDataFolder.
|       / 現在のデータフォルダーを PythonDataFolder として返す。
|
| ----------------------------------------------------------------------
| Static methods inherited from pybind11_builtins.pybind11_object:
| / pybind11_builtins.pybind11_object から継承した静的なメソッド:
|
| __new__(*args, **kwargs) class method of pybind11_builtins.pybind11_object
|       Create and return a new object. See help(type) for accurate signature.
|       / 新しいオブジェクトを作成して返す。正確なシグニチャは help(type) で確認。
```