CONTENTS

ビジュアルヘルプ – PythonFile

PythonFile コマンドのヘルプ

PythonFile [/Z] *file=fileStr* [, *var={pyVarStr, igorVar}*, *array={pyArrayStr, igorWave}*, *args=argListStr*]

PythonFile コマンドは、.py ファイル全体を先頭から末尾まで実行します。
注意：このコマンドはファイルをモジュールとしてインポートしません。
そのため、プログラマーはスクリプトを変更し、Python インタープリターを再起動せずに新しいコードをすぐに実行できます。

パラメーター

file=*fileStr*
実行する Python ファイルです。
これは、Python パス上のファイル名か、ファイルの完全なパスです。
fileStr に完全なパスを使う場合、パスの形成方法についてはヘルプ Path Separators を参照してください。
fileStr にはファイル拡張子（.py）を含める必要があります。
このキーワードは必須です。

var={*pyVarStr*, *igorVar*}
指定した Python 数値変数または文字列変数を、Igor 数値変数または文字列変数 igorVar に割り当てます。
igorVar は、グローバル変数へのフルパスには指定できません。
グローバル変数に割り当てるには、まず NVAR (数値変数) または SVAR (文字列変数) を使って、その変数をローカルで宣言する必要があります。

array={*pyArrayStr*, *igorWave*}
指定した Python 配列のようなオブジェクトを Igor ウェーブ igorWave に代入します。

igorWave はすでに存在している必要があり、ローカルウェーブ参照またはウェーブのフルパスです。
グローバルウェーブとフリーウェーブの両方がサポートされています。
詳細については、以下の「Python 変数を Igor に返す」を参照してください。

配列のようなオブジェクトには、NumPy 配列、リスト、タプル、または Python の組み込み配列（array.array）が含まれます。

args=*argListStr*
Python スクリプトに提供する引数のリストを指定する文字列リストです。

個々の引数は、空白 (タブまたはスペース) で区切ってください。
各引数は、sys.argv Python リストのエントリとして挿入され、Python スクリプト自体で解析されます。
文字列を返す Igor 式や関数も使用できます。
数値を含む文字列は、Python スクリプトで解析した後、数値型に戻す必要があります。

フラグ

/Z　　エラーを出力しません。
エラーメッセージは Python コンソールに出力され、S_PythonError に格納されます。

出力変数

V_flag　　エラーコード。エラーが発生しなかった場合は 0 に設定されます。

S_PythonError　　Python エラーが発生した場合、この文字列は Python のトレースバックメッセージに設定されます。

この文字列は Python エラーのみによって設定され、Igor 関連のエラーでは設定されません。
例えば、実行中の Python コードの構文エラーは S_PythonError を設定します。
ただし、var または array キーワードの使用中に変数またはウェーブが存在しない場合、それは Python エラーではなく Igor エラーであるため、この文字列は設定されません。

詳細

PythonFile は、次のように、システムコマンドプロンプトから Python ファイルを実行するのと同様の動作をします。

```
python myScript.py
```

そのため、PythonFile は Python のグローバル変数 __name__ を 「__main__」 に設定します。
これは、PythonFile によってスクリプトが実行された場合にのみ動作し、モジュールとしてインポートされた場合には動作しないコードのガードとして機能します。
ファイルのインポートと実行の違いを示す以下の例をご覧ください。

オブジェクトのライフタイム

コード実行中に作成されたグローバル Python 変数は、Igor インスタンスの存続期間中、永続的に保持されます。
その結果、以降の Python および PythonFile コマンドへの呼び出しでは、以前に作成された変数やインポートされたライブラリを自由に使用できます。
同様に、Python コンソールで実行されたコードも、インポートされたライブラリやコマンドから作成された変数にアクセスでき、逆もまた同様です。

Python スクリプトを実行する場合、メソッド内で作成された変数は、実行が終わるとガベージコレクションの対象となり、Igor に返すことができないことに注意してください。
Python 変数を Igor に返すには、その変数をグローバルスコープで作成する必要があります。

Python の値を Igor に返す

var または array キーワードが使われた場合、Igor は Python オブジェクトを既存の Igor オブジェクトに代入しようとします。
Igor は新しいオブジェクトを作成しません。
この処理を成功させるには、Python と Igor の型に互換性がある必要があります。
互換性のあるオブジェクトの説明については、ヘルプ Python to Igor Type Conversions を参照してください。
詳細な説明と例については、Python コマンドのヘルプの「Python 変数を Igor に返す」セクションを参照してください。

Python インタープリターの再起動

Igor を終了せずに新しい Igor エクスペリメントを開いても、Python インタープリターはリセットされません。
Python セッションを完全にクリーンな状態に戻す、あるいは Python 環境を変更するには、Igor を再起動する必要があります。

エラー処理

Python コードの実行中に発生したエラーは、実行時エラーとして Igor に中継されます。
エラーメッセージには、Python トレースバックメッセージが含まれます。
このメッセージは、Python コンソールにも表示されます。
他の Igor ランタイムエラーと同様、Python ランタイムエラーは try-catch-endtry 構文で捕捉できます。

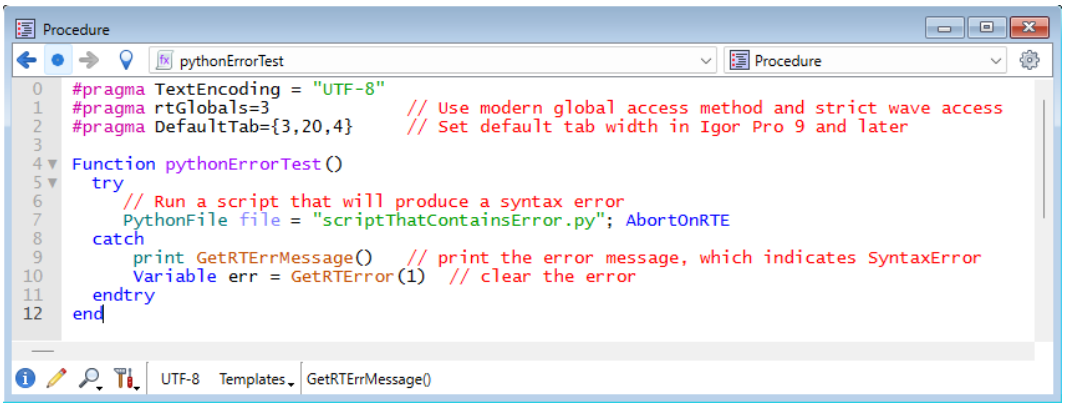
```
# Python
# scriptThatContainsError.py
print('this is missing a parentheses'  # produces a runtime error
```

```
// Igor 側
try
        // 構文エラーを生成するスクリプトを実行する
        PythonFile file = "scriptThatContainsError.py"; AbortOnRTE
catch
        print GetRTErrMessage()      // SyntaxError を示すエラーメッセージを出力する
        Variable err = GetRTError(1)        // エラーをクリアする
endtry
```

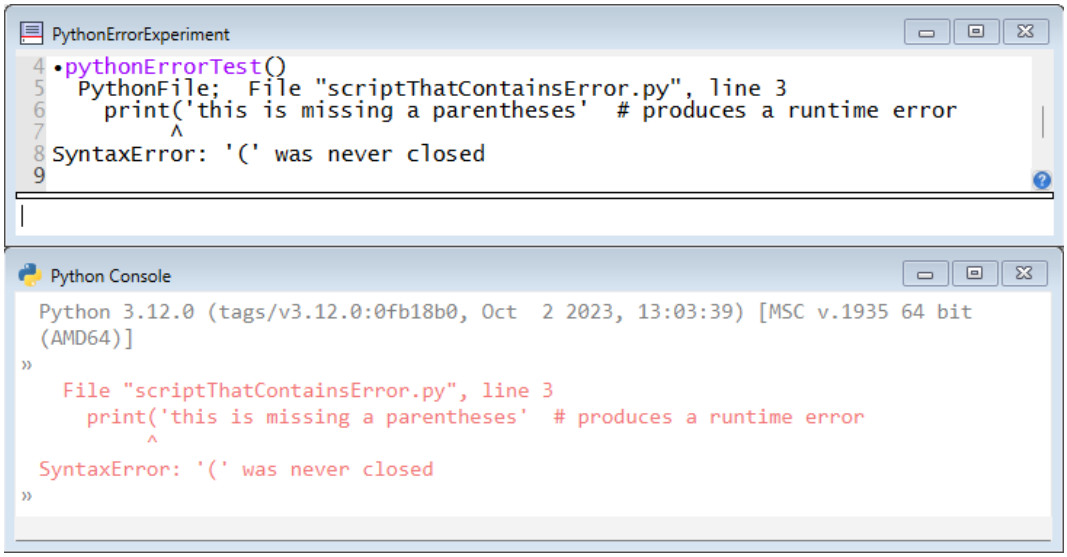
Python の実行中に発生したエラーは、Python コンソールに記録され、また S_PythonError 文字列にも保存されます。

次のようにして、確かめることができます。

1. 任意のエディターで上記の 3 行の Python のコードを作成し、Igor Pro 10 フォルダーの Python Scripts フォルダーにファイル名 scriptThatContainsError.py で保存します。
2. プロシージャウィンドウに次のような関数（ pythonErrorTest() ）を定義して、コンパイルします。



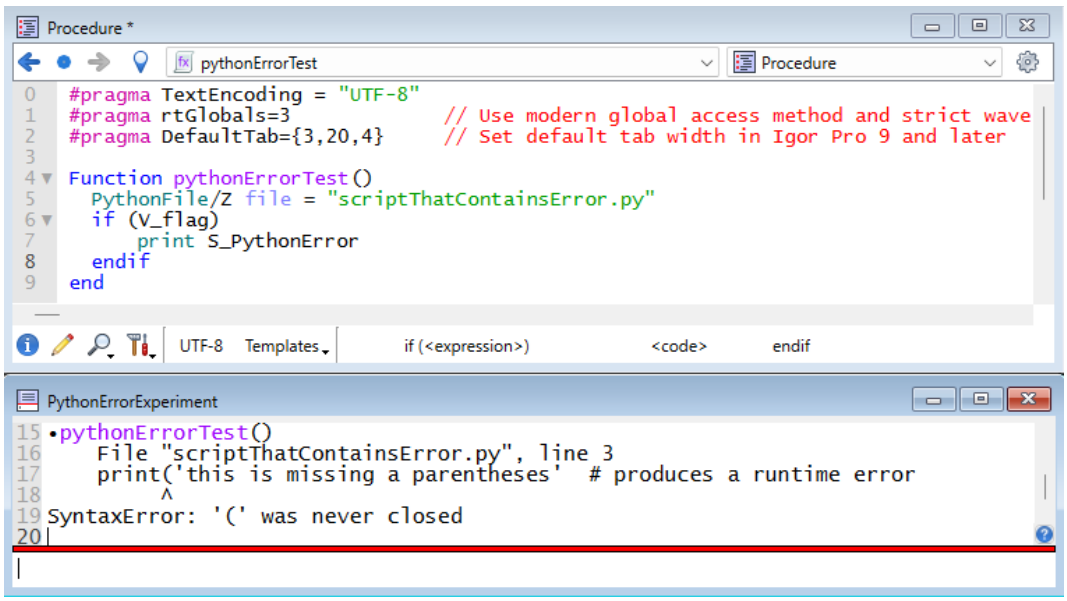
3. コマンドウィンドウで
   `pythonErrorTest()`
   を実行します。
4. Python コンソールに出力されたエラーが Igor の履歴エリアにも表示され、S_PythonError 文字列に保存されます。



エラーが発生した場合は、V_flag に 0 以外のエラーコードが設定されます。
これを使って、Python からトレースバックメッセージを取得することができます。
例えば：

```
// 構文エラーを生成するスクリプトを実行する
PythonFile/Z file = "scriptThatContainsError.py"
if (V_flag)
        print S_PythonError
endif
```

プロシージャウィンドウの関数 pythonErrorTest() を書き換えて、コマンドウィンドウで再度実行します。



例

引数付きで Python ファイルを実行する：

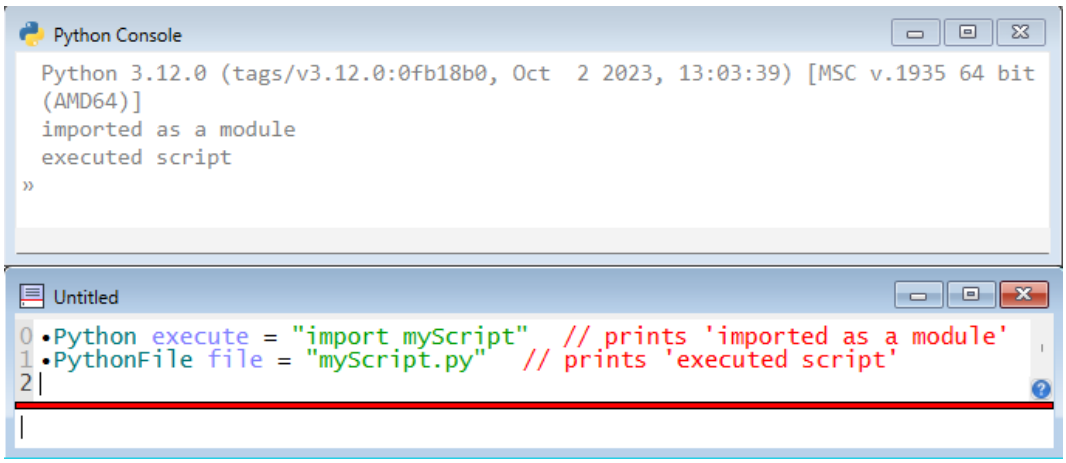
```
# Python
# myScript.py
import sys

def doAnalysis(arg):
  # 分析ルーチンを実行...
  return result = arg * 2

if __name__ == '__main__':
  arg1 = float(sys.argv[1])          # 引数を文字列から浮動小数点数に変換する
  result = doAnalysis(arg1)          # 'result' はグローバル変数
```

```
// Igor 側
Variable result
PythonFile file = "myScript.py", var = {"result", result}, args = "3.141"
Print result                          // 6.282 を出力
```

Python ファイルのインポートと実行の比較：

```
# Python
# myScript.py
if __name__ == '__main__':
  print('executed script')
else:
  print('imported as a module')
```

```
// Igor 側
Python execute = "import myScript"        // 'imported as a module' を出力
PythonFile file = "myScript.py"           // 'executed script' を出力
```



参照

Python、PythonEnv、ヘルプ Python Overview、Python Module Reference