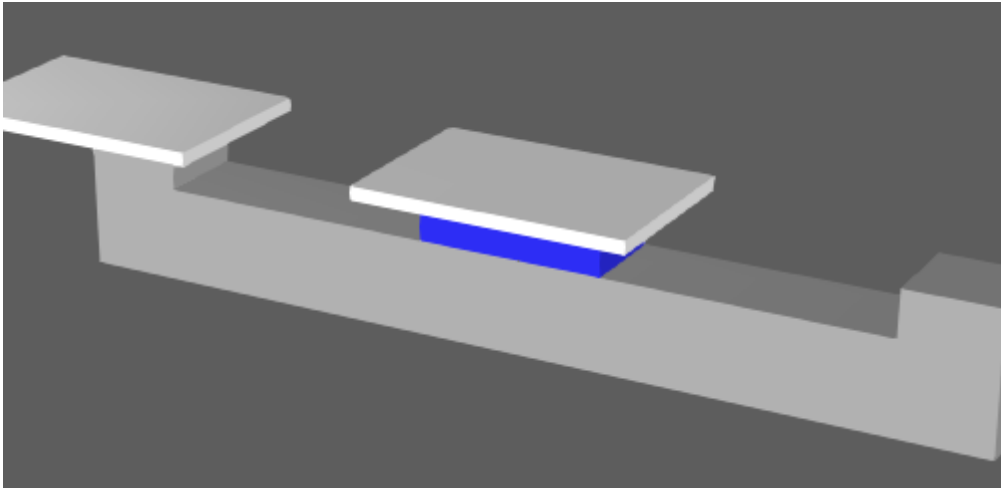


Error Management

Visual Components 4.0 | Version: February 23, 2017



When using Python API to write application and component scripts you may encounter errors and exceptions. In order for your script to work, you will need to manage and resolve errors and handle exceptions. Errors may occur at the time of compilation or execution of the script. Your main source of feedback about these and other issues is the Output panel.

In this tutorial you learn how to:

- Manage compile and runtime errors
- Resolve syntax and semantic errors related to Python
- Handle exceptions

For more information about errors and exceptions in Python, go to

<https://docs.python.org/2/tutorial/errors.html>

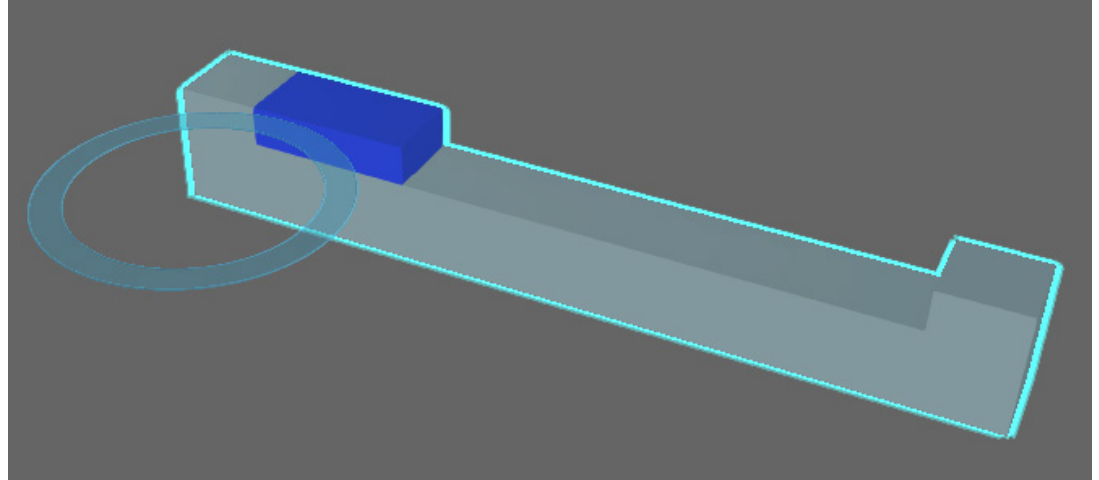
Support
support@visualcomponents.com

Community
community.visualcomponents.net

Compile Error

A compile error may occur when you compile a Python Script behavior.

1. Open the **ErrorManagementExample.vcmx** file for this tutorial.



2. In the Output panel, notice that you get feedback about a syntax error.

```
File "Error Management Example::Script", line 13
```

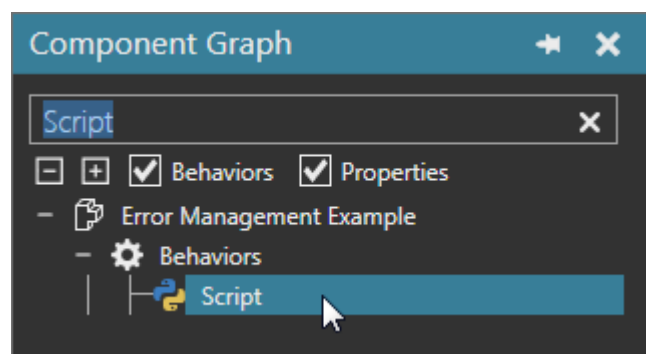
```
whie True:
```

```
  ^
```

```
SyntaxError: invalid syntax
```

This is a compile error that occurred when the component was loaded in the 3D world. The error was found in a Python Script named "Script" on line 13. The script itself is contained in the Error Management Example file/component. The syntax error is a misspelling of the word "while" in the script.

3. In the Output panel, right-click, and then click **Clear**.
4. Click the **Modeling** tab, and then in the Component Graph panel, do a search for "Script" and then double-click the found **Script** behavior to access its editor.



5. In the editor, go to line 13, and then change "whie" to "while" and then compile the script.



```
1 Compile from vcScript import *
2
3 def OnRun():
4     comp = GetComponent()
5     Servo = comp.findBehaviour("ServoController")
6     PartEnterSignal = comp.findBehaviour("ComponentEnteringSignal")
7     ServoSensorSignal = comp.findBehaviour("ServoSensorSignal")
8     ServoPath = comp.findBehaviour("ServoPath")
9
10    object = comp.findBehaviour("Twist_DeleteMe")
11
12
13    while True:
14        ServoPath.Enabled = False
15        Servo.moveJoint(0,0.0)
16
```

6. Check the Output panel for any errors. The compiler should not find any errors nor print any feedback.

Runtime Error

A runtime error may occur when the main application executes a Python Script behavior during a simulation.

1. Minimize the Script editor, and then run the simulation.
2. In the Output panel, notice that you get feedback about a name error.

```
Traceback (most recent call last):  
  File "Error Management Example::Script", line 27, in <lambda>  
NameError: global name 'ServoSensorSigna' is not defined
```

This is a runtime error that occurred during a simulation in the same component and script. The name error was found on line 27 and is a typo of the "ServoSensorSignal" variable declared on line 7.

3. Reset the simulation, and then clear the Output panel.

IMPORTANT! You cannot recompile a Python Script behavior during a simulation. You will need to first reset the simulation, and then recompile the script.

4. Restore the Script editor, and then on line 27, change "ServoSensorSigna" to "ServoSensorSignal" and then compile the script.

```
26 ServoPath.Enabled = True  
27 condition(lambda: ServoSensorSignal.Value != None )  
28 ServoSensorSignal.Value = None
```

5. Run the simulation, verify no errors are printed in the Output panel, and then reset the simulation.

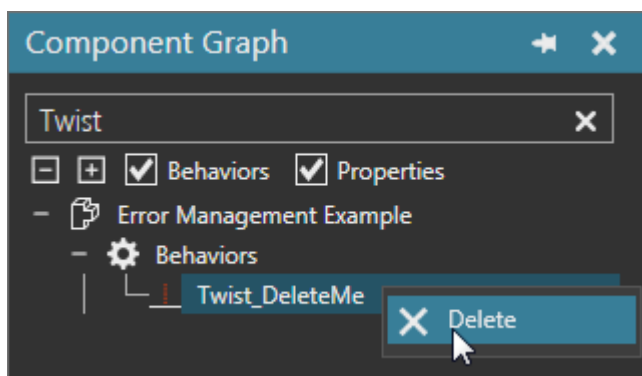
Exception Handling

There are several ways to manage errors. The previous sections were ad hoc approaches that dealt with errors as they occurred, for example at script compilation and execution. There is a way to deal with errors by writing scripts that take into account potential errors. For example, lines 19 to 24 in the Script editor show a try statement with an except clause.

```
try:  
    object.SampleTime = 0.1  
    Servo.moveJoint(0,100.0)  
    Servo.moveJoint(0,0.0)  
except:  
    print "Twist_DeleteMe behaviour not accessible - managed using exception"
```

If an error occurs in the try statement, an exception is raised and handled by the except clause.

1. Run the simulation. Since there are no errors, the code in the try statement is executed, which is why the platform of the component moves back and forth before picking up a product.
2. Stop the simulation, but do not reset it. Why? Since the script has a handler for an exception, you can make some changes without any serious impact to the simulation.
3. In the Component Graph panel, do a search for "Twist" and then delete the found behavior. It should be the Raycast Sensor behavior referenced by the "object" variable in the try statement.



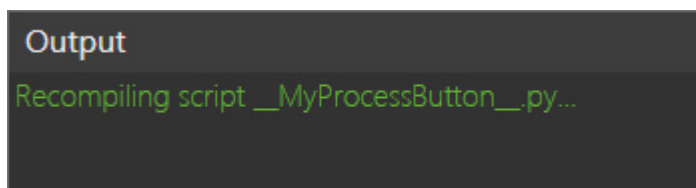
4. Continue the simulation, verify the exception handler prints feedback in the Output panel, the platform no longer moves back and forth before picking up a product, and a potential error was handled without having to reset the simulation.

```
Output  
Twist_DeleteMe behaviour not accessible - managed using exception
```

Add-ons

When you start a Visual Components 4.0 product, the main application looks in the My Commands folder of your Visual Components documents for `__init__` files. These files are used to register and load Python add-ons. If an error is found in an `__init__` file, the main application does not register the add-on associated with that file. That is, you cannot use the add-on.

An error may occur when you use a Python add-on. Generally, this is because of an error in the command file of that add-on. You can edit and save a command file without having to restart a Visual Components 4.0 product. If you update a command file, the main application will automatically recompile the command file the next time you use the Python add-on associated with that file.



```
Output
Recompiling script __MyProcessButton__.py...
```

Review

In this tutorial you learned how to manage errors and handle exceptions in a component script. You know the difference between compile and runtime errors and how to manage both with the aid of feedback printed in the Output panel. You can use this same knowledge when managing application scripts. You know that errors might occur when the main application initializes an add-on or when you are using the add-on.