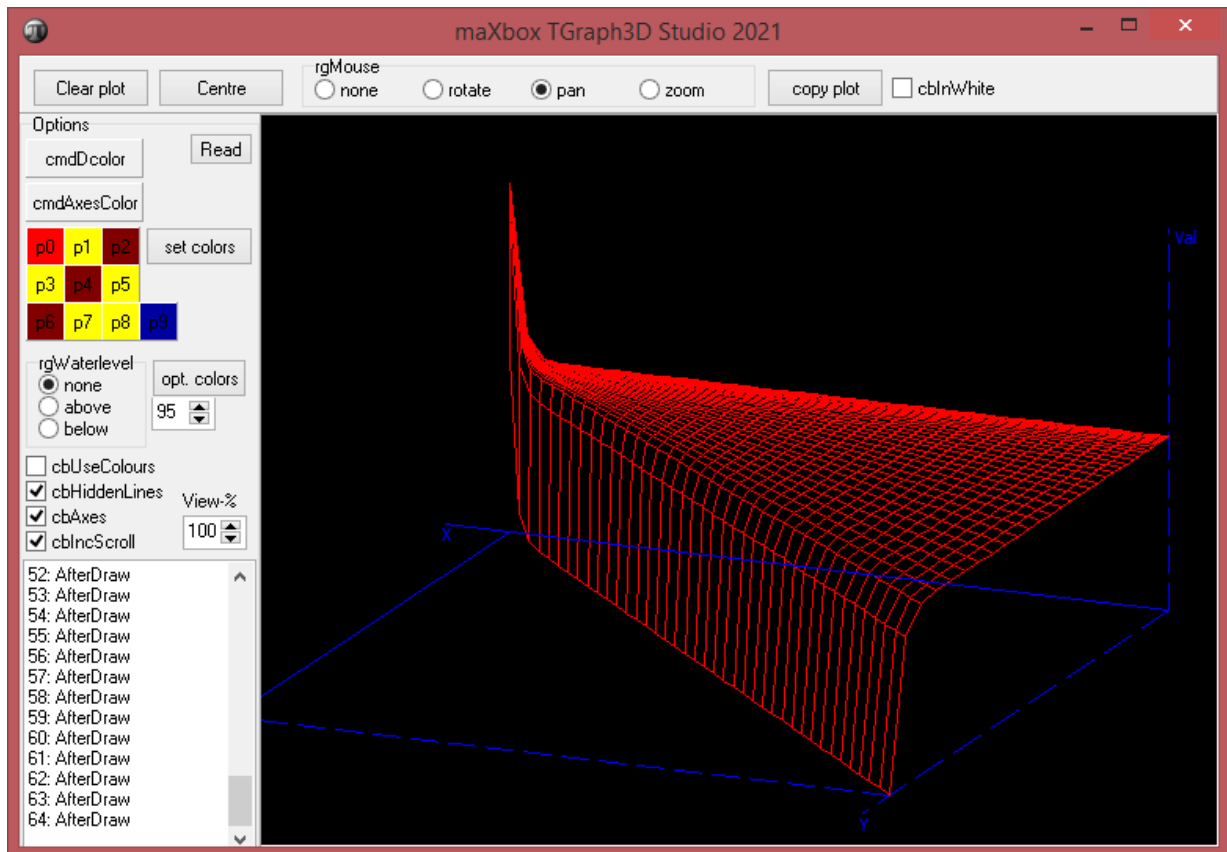


Code Blog

Python4Delphi

 [maxbox4](#)  [June 9, 2021](#)  [Code](#)  [delphi, pascal, python](#)  [Edit](#)



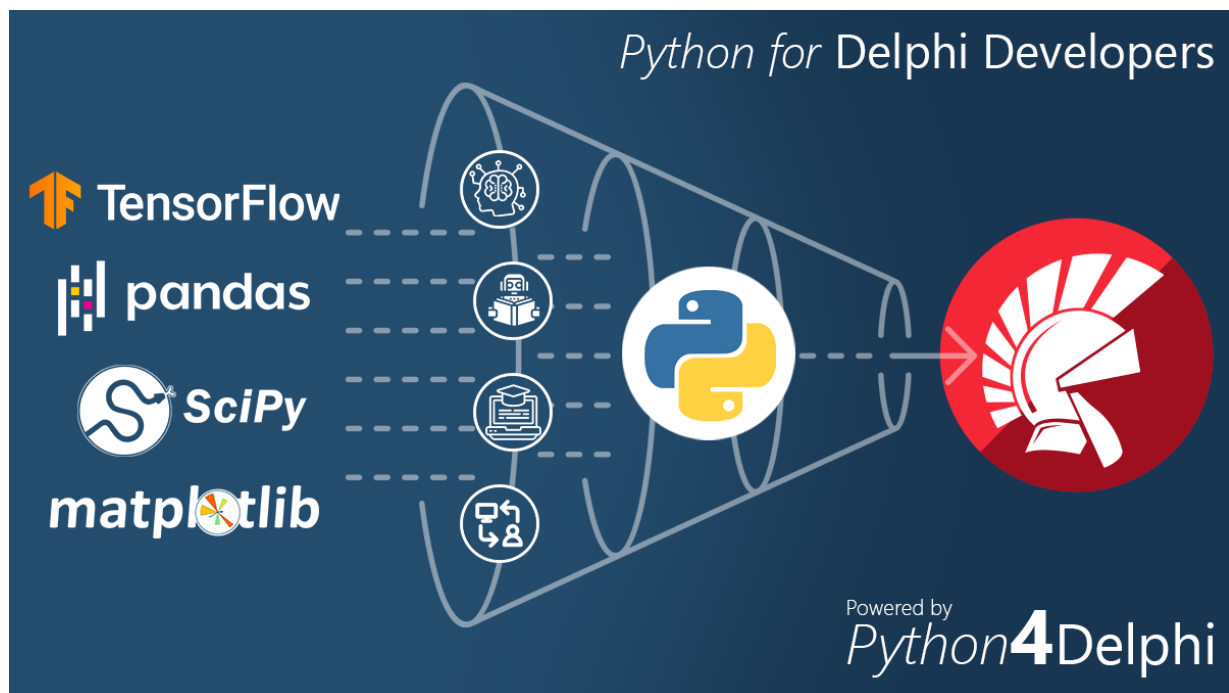
In a future world you can decide belonging to SkyNet or Darknet, but you can not find the difference between an Android or Avatar cause humans doesn't exist anymore.

Max Kleiner

Python for **Delphi** (P4D) is a set of free components that wrap up the Python DLL into Delphi and Lazarus (FPC). A DLL could be for example the 'python37.dll'. They let you almost easily execute Python scripts, create new Python modules and new Python types. You can create Python extensions as DLLs and much more like scripting or automating your console. P4D provides different levels of functionality:

- Low-level access to the Python API
- High-level bi-directional interaction with Python
- Access to Python objects using Delphi custom variants (VarPyth.pas)
- Wrapping of Delphi objects for use in Python scripts using RTTI (WrapDelphi.pas)
- Creating Python extension modules with Delphi classes, records and functions
- Generate Scripts in maXbox from a Python engine.

P4D makes it, after some (tough) studies, very easy to use Python as a scripting language for Delphi applications. It also comes with an extensive range of demos and useful (video) tutorials.



P4D

So a very first simple approach is to call the Python dll without a wrapper or mapper e.g. call the copyright function:

```

1 | /if fileExistst(PYDLLPATH+ 'python37.dll';
2 |   function getCopyRight: PChar;
3 |     external 'Py_GetCopyright@C:\maXbox\EKON25\python37.dll

```

Then we call the function with a pre-test:

```

1  function IsDLLOnSystem(DLLName:string): Boolean;
2  var ret: integer;
3      good: boolean;
4  begin
5      ret:= LoadLibrary(pchar(DLLNAME));
6      Good:= ret>0;
7      if good then FreeLibrary(ret);
8      result:= Good;
9  end;
10
11  if isDLLOnSystem(PYDLLPATH+PYDLLNAME) then begin
12      showmessage('py dll available');
13      writeln(getCopyRight)
14  end;

```

Copyright (c) 2001-2019 Python Software Foundation.

All Rights Reserved.

Copyright (c) 2000 BeOpen.com.

All Rights Reserved.

Copyright (c) 1995-2001 Corporation for National Research Initiatives.

All Rights Reserved.

Copyright (c) 1991-1995 Stichting Mathematisch Centrum, Amsterdam.

All Rights Reserved.

We also use to invoke a Python script as an embedding **const** and use the dll functionality of *Import('PyRun_SimpleString')*;

```

1  procedure InitSysPath;
2  var _path: PPyObject;
3  const Script =
4      'import sys' + sLineBreak+
5      'sys.executable = r"%s"' + sLineBreak+
6      'path = sys.path' + sLineBreak+
7      'for i in range(len(path)-1, -1, -1):' + sLineBreak+
8      '    if path[i].find("site-packages") > 0:' + sLineBreak
9      '        path.pop(i)' + sLineBreak+
10     'import site' + sLineBreak+
11     'site.main()' + sLineBreak+
12     'del sys, path, i, site';
13  begin
14      if VenvPythonExe <> '' then
15          ExecString(AnsiString(Format(Script, [VenvPythonExe]))
16          _path := PySys_GetObject('path');
17      if Assigned(FOnSysPathInit) then
18          FOnSysPathInit(Self, _path);
19  end;

```

How to use Python4Delphi

The best way to learn about how to use P4D is to try the extensive range of demos available. Also studying the unit tests for VarPyth and WrapDelphi can help understand what is possible with these two units and the main ***PythonEngine.pas***.

Lets start with the *VarPyth.pas* unit.

This allows you to use Python objects like COM automation objects, inside your Delphi source code. This is a replacement, bugfixed of the former *PythonAtom.pas* that uses the new custom variant types introduced since Delphi6.

You may use these Python Variants in expressions just as you would any other Variants or automation e.g.:

```
1  var
2    a: Integer; v: Variant;
3  begin
4    v:= VarPythonEval('2 ** 3');
5    a:= v;
```

The functions provided by this unit *VarPyth.pas* are largely self-explanatory.

What about the *WrapDelphi.pas* unit?

You can use P4D to create Python extension modules too that expose your own classes and functions to the Python interpreter. You may package your extension with setuptools and distribute it through **PyPi**. So if you have an existing object or unit in Delphi that you'd like to use in Python but you don't want to modify that object to make it Python-aware, then you can wrap that object just for the purposes of supplying it to Python with a package.

Using *TPyDelphiObject* you can wrap any Delphi object exposing published properties and methods. Note that the conditional defines TYPEINFO and METHODINFO need to be on. As an example, if you have an existing Delphi class simply called TRGBColor:

```
1  TRGBColor = class
2    private
3      fRed, fGreen, fBlue: Integer;
4    public
5      property Red: read fRed write fRed;
6      property Green: read fGreen write fGreen;
7      property Blue: read fBlue write fBlue;
8  end;
```

You want to use Color within some Python code but you don't want to change anything about the class. So you make a wrapper inherited from **TPyObject** that provides some very basic services, such as getting and setting attributes of a Color, and getting a string representation:

```

1  TPyColor = class(TPyObject)
2      private
3          fColor: TRGBColor;
4      public
5          constructor Create( APythonType: TPythonType ); override
6          // Py Basic services
7          function GetAttr(key: PChar): PPyObject; override;
8          function SetAttr(key: PChar; value: PPyObject): Integer
9          function Repr: PPyObject; override;
10     end;
```

The project in the subdirectory Delphi generates a Python extension module (a DLL with extension “pyd” in Windows) that allows you to create user interfaces using Delphi from within Python. The whole VCL (almost and maybe) is wrapped with a few lines of code! The small demo *TestApp.py* gives you a flavour of what is possible. The machinery by which this is achieved is the WrapDelphi unit.

The subdirectory DemoModule demonstrates how to create Python extension modules using Delphi, that allow you to use in Python, functions defined in Delphi. Compile the project and run test.py from the command prompt (e.g. py test.py). The generated pyd file should be in the same directory as the Python file. This project should be easily adapted to use with **Lazarus** and **FPC**.

The subdirectory RttiModule contains a project that does the same as the DemoModule, but using extended RTTI to export Delphi functions. This currently is not supported by FPC.

The unit ***PythonEngine.pas*** is the main core-unit of the framework. You are responsible for creating one and only one TPythonEngine. Usually you just drop it on your main form. Most of the Python/C API is presented as member functions of the engine.

```

1  type
2      TPythonVersionProp = record
3          DllName      : string;
4          RegVersion   : string;
5          APIVersion   : Integer;
6      end;
7
8      ...
9      Py_BuildValue      := Import('Py_BuildValue');
10     Py_Initialize      := Import('Py_Initialize');
11     PyModule_GetDict   := Import('PyModule_GetDict');
12     PyObject_Str       := Import('PyObject_Str');
13     PyRun_String        := Import('PyRun_String');
14     PyRun_SimpleString  := Import('PyRun_SimpleString');
15     PyDict_GetItemString := Import('PyDict_GetItemString');
16     PySys_SetArgv       := Import('PySys_SetArgv');
17     Py_Exit             := Import('Py_Exit');
18     ...

```

Let's take a last look at the functionality of *PyRun_SimpleString* mentioned first within the const script.

http://www.softwareschule.ch/examples/1016_newsfeed_sentiment_integrate2.txt

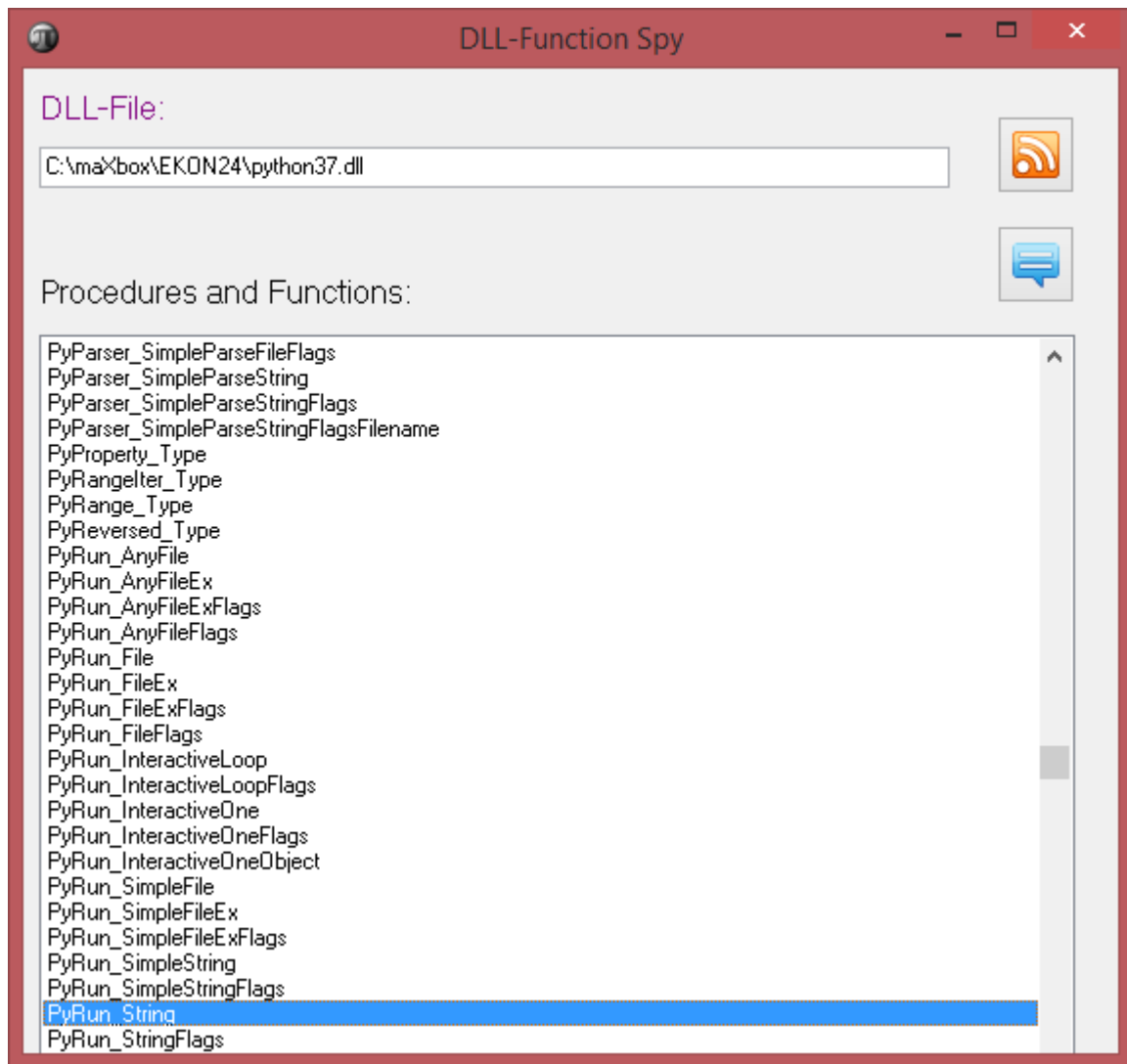
PyRun_SimpleString: function(str: PAnsiChar): Integer; cdecl;

We see that we have to pass a PAnsiChar in cdecl convention and map to *ExecString* (PyRun_String):

```

1  procedure TPythonEngine.ExecString(const command : AnsiString
2  begin
3      Py_XDECREF(Run_CommandAsObject(command, file_input));
4  end;

```



DLL Spy

Conclusion

The P4D library provides a bidirectional bridge between Delphi and Python. It allows Delphi applications to access Python modules and run Python scripts. On the other side it makes Delphi/Lazarus objects, records, interfaces, and classes accessible to Python, giving Python the ability to use this methods.

Before you try the demos please see the Wiki topic “How Python for Delphi finds your Python distribution” at

<https://github.com/pyscripter/python4delphi/wiki/FindingPython>

You will need to adjust the demos accordingly, to successfully load the Python distribution that you have installed in your PC, e.g. C:\Users\max\AppData\Local\Programs\Python\Python36\.

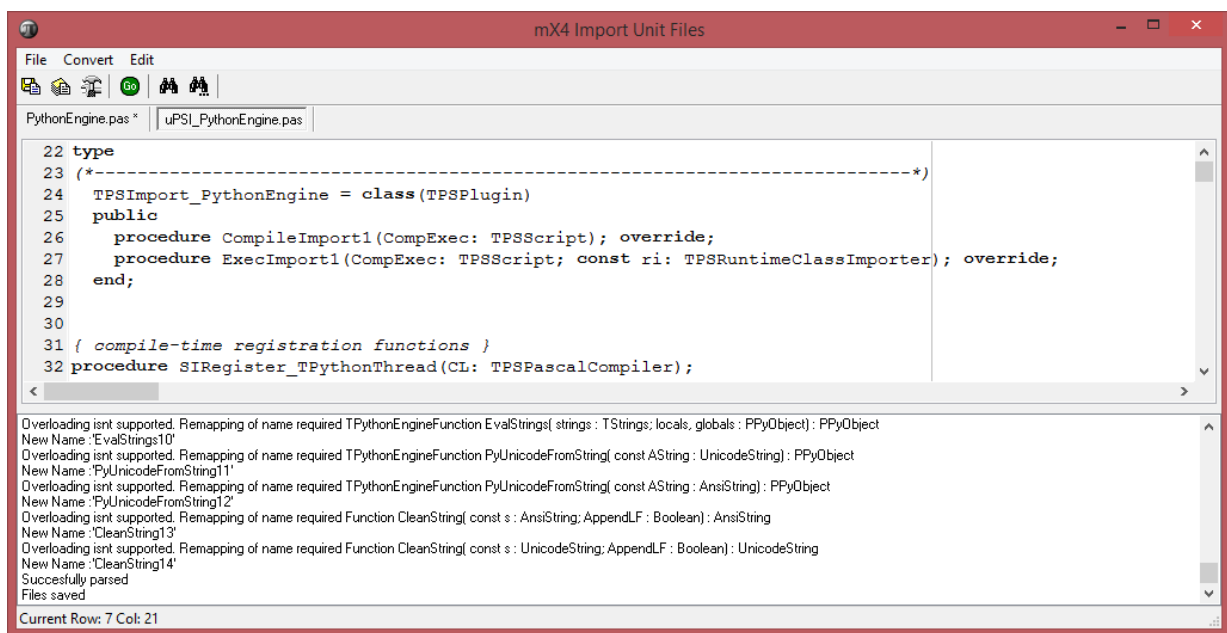
PythonEngine: The core of Python for Delphi. Provides the Python

API with dll mapper and runtime configuration.

VarPyth: VarPyth wraps Python types as Delphi custom variants.

WrapDelphi: Uses RTTI (in a DLL) so you can use Delphi objects from Python without writing individual wrapper classes or methods.

TPythonGUIInputOutput: Provides a Python console you can drop on a form and execute a Python script from a memo.



mX4 Conversion

The TPythonEngine class is the single global engine block shared by all Python code. **VarPyth** wraps Python types and vars as Delphi custom variants. **VarPyth** requires at least Delphi v6.

This Tutorials folder contains text and video, webinar tutorials accompanied with slides and demo source code. Next time I'll show a few bidirectional demos with implementation details.



Wiki P4D topics

- [Installation](#)
- [Supported Platforms](#)
- [How Python for Delphi finds your Python distribution](#) (read before trying the demos)

Learn about Python for Delphi

- [Tutorials](#)
 - [Demos](#)
- <https://github.com/maxkleiner/python4delphi>

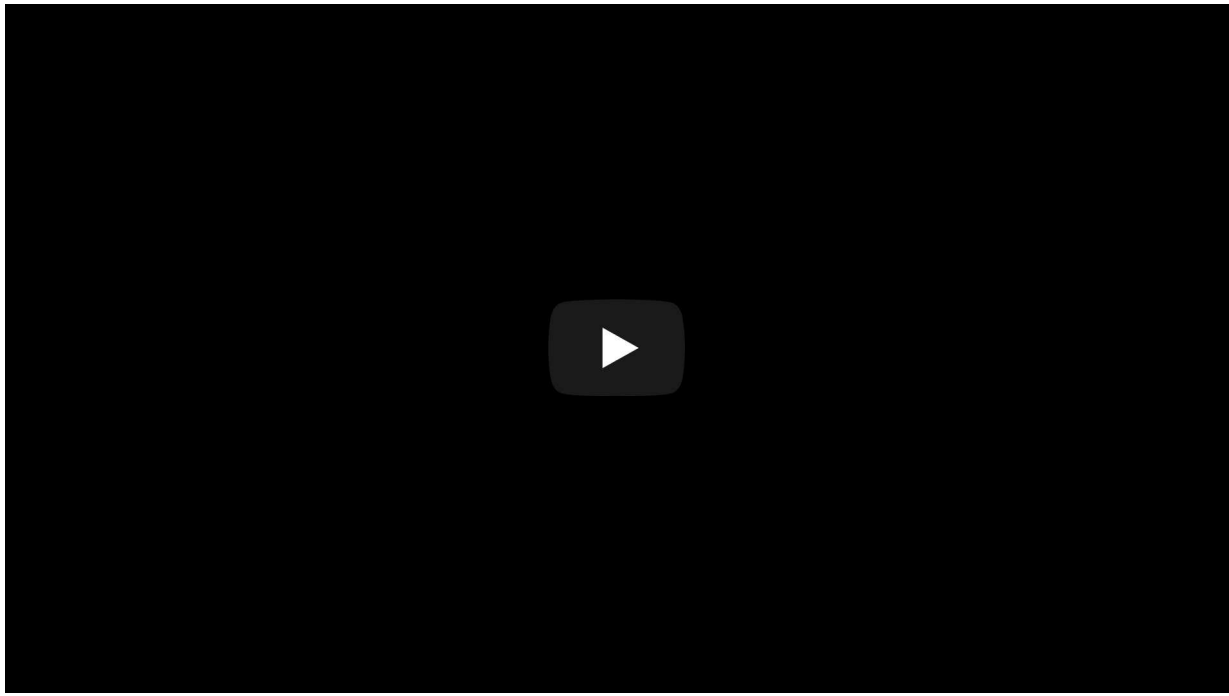
Dealing with internals of Python means also you get the original stack-trace errors back like (*TPythonEngine.RaiseError*):

```
File "C:\Users\max\AppData\Local\Programs\Python\Python36
\lib\site-packages\pandas\core\indexing.py", line 1304, in
_validate_read_indexer raise KeyError(f"{not_found} not in
index") KeyError: "['id', 'links', 'published_parsed',
'published', 'title_detail', 'guidislink', 'link',
'summary_detail'] not in index"
```

The script can be found:

http://www.softwareschule.ch/examples/1016_newsfeed_sentiment_integrate2.txt

Ref Video:



<https://github.com/pyscripter/python4delphi/wiki/FindingPython>

You will need to adjust the demos accordingly, to successfully load the Python distribution that you have installed on your computer.

Doc: <https://maxbox4.wordpress.com>

Appendix: Catalog of the Demos:

{*-----*)

Demo01 A simple Python evaluator
Demo02 Evaluate a Python expression
Demo03 Defining Python/Delphi vars
Demo04 Defining Python/Delphi vars (advanced case)
Demo05 Defining a new Module
Demo06 Defining a new Type
Demo07 Using Delphi methods as Python functions
Demo08 Using Delphi classes for new Python types
Demo09 Making a Python module as a Dll
Demo10_FireDAC Database demo using FireDAC
Demo11 Using Threads inside Python
Demo16 Using a TDelphiVar or Module methods
Demo17 Using variant arrays of 2 dimensions
Demo19 C++ Builder: this is a replicate of the Delphi Demo05
Demo20 C++ Builder: this is a replicate of the Delphi Demo08
Demo21 Using Events in TPythonModule or TPythonType
Demo22 Using Threading, Windows Console and Command line arguments
Demo23 Using Threading and Delphi log window
Demo25 Using VarPyth.pas
Demo26 Demo08 revisited to allow the new Python type to be subclassed
Demo27 Container indexing
Demo28 Iterator
Demo29 Using Python Imaging Library (PIL)
Demo30 Using Named Parameters
Demo31 Using WrapDelphi to access Delphi Form attributes
Demo32 Demo08 revisited using WrapDelphi
Demo33 Using Threads inside Python
Demo34 Dynamically creating, destroying and recreating PythonEngine.

The screenshot displays the Academia.edu profile of Max Kleiner. At the top, a blue banner advertises a 50% discount on Academia Premium, with a countdown timer showing 47 hours, 59 minutes, and 24 seconds remaining. The profile header includes a search bar, navigation tabs for Researcher, Courses, Mentions, and Audience, and a Premium upgrade button. The profile itself features a classical statue head as a profile picture, the name Max Kleiner, and a bio mentioning his affiliation with Bern University and his work on the maXbox scripter studio. A 'READERS UPDATE' pop-up is overlaid on the profile, indicating that an Arduino researcher has read his paper 'Webofthings2013'. Below the profile, it states that 8,201 papers mention Max Kleiner. A Credit Suisse advertisement is visible at the bottom right.

Acad

Programmers when they don't understand someone else's code!



Stonecold

Advertisements

Occasionally, some of your visitors may see an advertisement here, as well as a [Privacy & Cookies banner](#) at the bottom of the page. You can hide ads completely by upgrading to one of our paid plans.

[UPGRADE NOW](#)[DISMISS MESSAGE](#)

Share this:[Customize buttons](#)

Be the first to like this.

Related[The OMA Model](#)

September 12, 2020

In "Machine Learning"

[JSON Code](#)

March 5, 2021

In "Code"

[Energy Star Predictor](#)

January 24, 2020

In "Machine Learning"

 [maxbox4](#)  [June 9, 2021](#)  [Code](#)  [delphi, pascal, python](#)  [Edit](#)

Published by maxbox4

Code till the End [View more posts](#)

One thought on “Python4Delphi”

[maxbox4](#)**[June 9, 2021 at 3:44 pm](#)**

For more insights I did an Overview of the Project in a 7 pages handout.

Overview of P4D: http://www.softwareschule.ch/download/maxbox_starter86.pdf

 Like[Reply](#)

Leave a Reply

Enter your comment here...

Code Blog, Website Powered by WordPress.com.