Introduction to Python GUI Development

with

Delphi for Python

Part 1: Delphi VCL for Python

Jim McKeeth
Chief Developer Advocate
Embarcadero Technologies
jim.mckeeth@embarcadero.com
@JimMcKeeth

embarcadero®

Slides, links & replay: blogs.embarcadero.com/?p=128183

Agenda

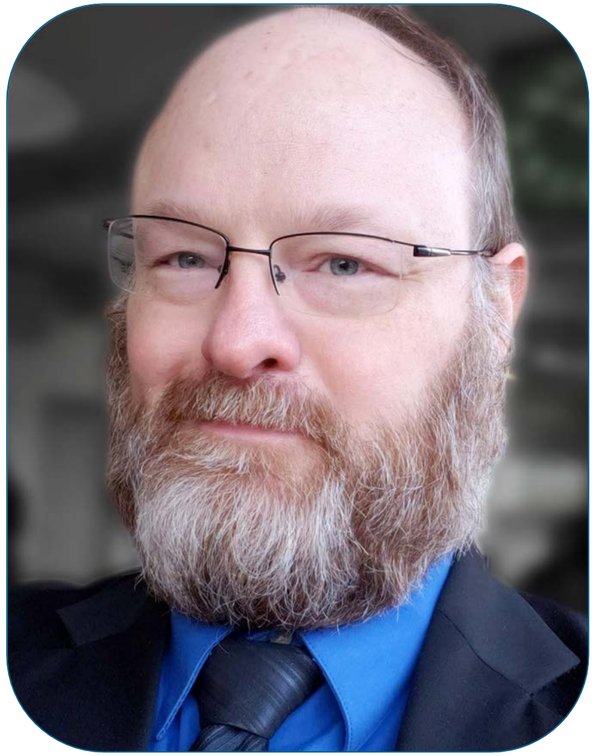
- Introduction to Delphi VCL & FMX
- Architecture and platforms
- Installing and using Delphi VCL for Python
- Demonstrations & Code
- Going Beyond - Mixing Delphi & Python
- More information, Next steps, Q&A

embarcadero®

Slides, links & replay: blogs.embarcadero.com/?p=128183

About Jim McKeeth

- Chief Developer Advocate & Engineer for Embarcadero
- Long time software developer
  - Delphi, C/C++, Python, Java, JavaScript, Ruby, etc.
- Invented and patented pattern and swipe to unlock
  - e.g. US Patents # 8352745 & 6766456
- Built thought controlled drone with Google Glass and wireless EEG headset
- Contributor to Internet of Things and Data Analytics Handbook
- Blogger, podcaster, conference speaker, webinar host, etc.
- Twitter, TikTok, YouTube, etc. @JimMcKeeth



Target Audience

1. Python devs who want a nice GUI
2. Python devs curious about Delphi
3. Delphi devs who want to use Python
4. Delphi devs curious about what's new
5. Other devs curious about Delphi & Python

I do my best not to assume too much familiarity with either Delphi or Python, while still including code and technical details.

It’s not a Competition

- Developers have multiple tools on their workbench
- It is about finding the right tool for each task
- Having specialized tools for different tasks doesn’t detract from favorite tools
- You can always find a specific task that another tool is better for, but no one tool is that best for all tasks.

What is Delphi for Python?

- Set of free Python modules bringing Delphi's GUI libraries to Python developers
  - Mature, feature rich, native & cross-platform
  - *Does not require Delphi to use*
- Based on the open source **Python4Delphi** (same technology that powers *PyScripter*)
- Available today on GitHub & PyPi
  - Currently in beta, but ready for use
- **DelphiVCL for Python** supports Windows (32-bit & 64-bit)
- **DelphiFMX for Python** adds support for Linux 64-bit, Android, & Mac OS
- Part of a bidirectional bridge between Delphi and Python

Introduction to Delphi VCL & FMX

For anyone new to Delphi with comparisons to Python

Timeline for Delphi & Python


Niklaus Wirth


Anders Hejlsberg


Guido van Rossum

1970 — Pascal

Created by Niklaus Wirth and Influenced by ALGOL 68.

1983 — Turbo Pascal

Developed by Anders Hejlsberg and promoted by Borland. The definitive Pascal.

1987 — ABC Programming Language

An imperative general-purpose programming language influenced by ALGOL 68. Guido van Rossum worked on ABC system for several years.

1991 — Python Introduced

Developed by Guido van Rossum as the successor to ABC.

1995 — Delphi Introduced

The successor to Turbo Pascal, includes the VCL. Using an evolved version of Object Pascal.

2000 — Python 2.0

Adding list comprehensions, reference counting, & cycle-detecting garbage collection

2008 — Python 3.0

A Major revision with many improvements.

2011 — Delphi XE adds Mac OS

Introducing FireMonkey, adding Windows 64-bit and Mac OS as natively supported platforms.

2013 — Delphi XE4 & XE5

Compiles to native ARM supporting, iPhone, iPad, and Android.

2017 — Delphi 10.2 *Tokyo*

Support for natively compiled x86 64-bit Linux.

2018 — Guido van Rossum's Permanent Vacation

Stepping down as Python's benevolent dictator for life (BDFL).

Today — Delphi for Python

Brings Delphi's VCL & FMX frameworks to Python

Photo of Guido by D. Stroud,et al https://w.wiki/4fSN
Photo for Wirth by Tyomitch https://w.wiki/4fSS
Photo of Hejlsberg by DBegley https://w.wiki/4fST

Delphi's DNA

1. **Developer productivity** - Really the main goal is getting things done quickly
2. **Maintainability** - Code is easy to read and understand with good encapsulation
3. **Fast compiled native apps** - Compiles fast, and native applications run fast
4. **Database access** - Always includes a rich set of database access components
5. **Platform API access** - You don't need to call platform APIs, but can if you want
6. **Property-Method-Event** - General model for working with components
7. **Visual designers** - WYSIWYG with drag and drop interface
8. **Reliable applications** - Exception handling and component owner model
9. **Backwards compatibility** - Even with all the updates most code is compatible
10. **Rich component ecosystem** - There is usually a component for everything

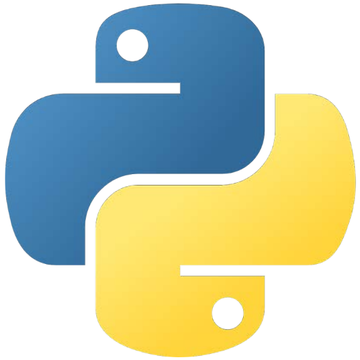
Zen of Python

19 "guiding principles" that influence the design of the Python programming language

1. *Beautiful is better than ugly.*
2. *Explicit is better than implicit.*
3. Simple is better than complex.
4. Complex is better than complicated.
5. Flat is better than nested.
6. Sparse is better than dense.
7. *Readability counts.*
8. Special cases aren't special enough to break the rules.
9. Although practicality beats purity.
10. Errors should never pass silently.
11. Unless explicitly silenced.
12. In the face of ambiguity, refuse the temptation to guess.
13. There should be one – and preferably only one – obvious way to do it
14. Although that way may not be obvious at first unless you're Dutch.
15. *Now is better than never.*
16. Although never is often better than right now.
17. *If the implementation is hard to explain, it's a bad idea.*
18. If the implementation is easy to explain, it may be a good idea.
19. Namespaces are one honking great idea – let's do more of those!

by Tim Peters, 1999, on the Python mailing list

Comparison



Introduced in 1995	↔	Introduced in 1991
Original architect is Danish	↔	Original author is Dutch
Commercial license (with free CE)	↔	Open source
Statically-typed (Explicit, or inferred)	↔	Dynamically-typed
Manual memory management with ownership model	↔	Garbage Collected (reference counting & cycle-detecting)
Compiled (with interpreted options*)	↔	Interpreted (optional JIT or compiler)
Popular in line of business applications	↔	Popular in research and prototyping

Language encourages best practices & readability

Includes comprehensive reusable libraries

Rich ecosystem of developers and libraries

Supports structured, procedural, functional, and OOP

Fanatical community of developers

*Interpreted Delphi options are via 3rd parties

Three Levels of Development

Universal to all development tools

Components & Designers (Very Visual)

Low Code

Common Libraries

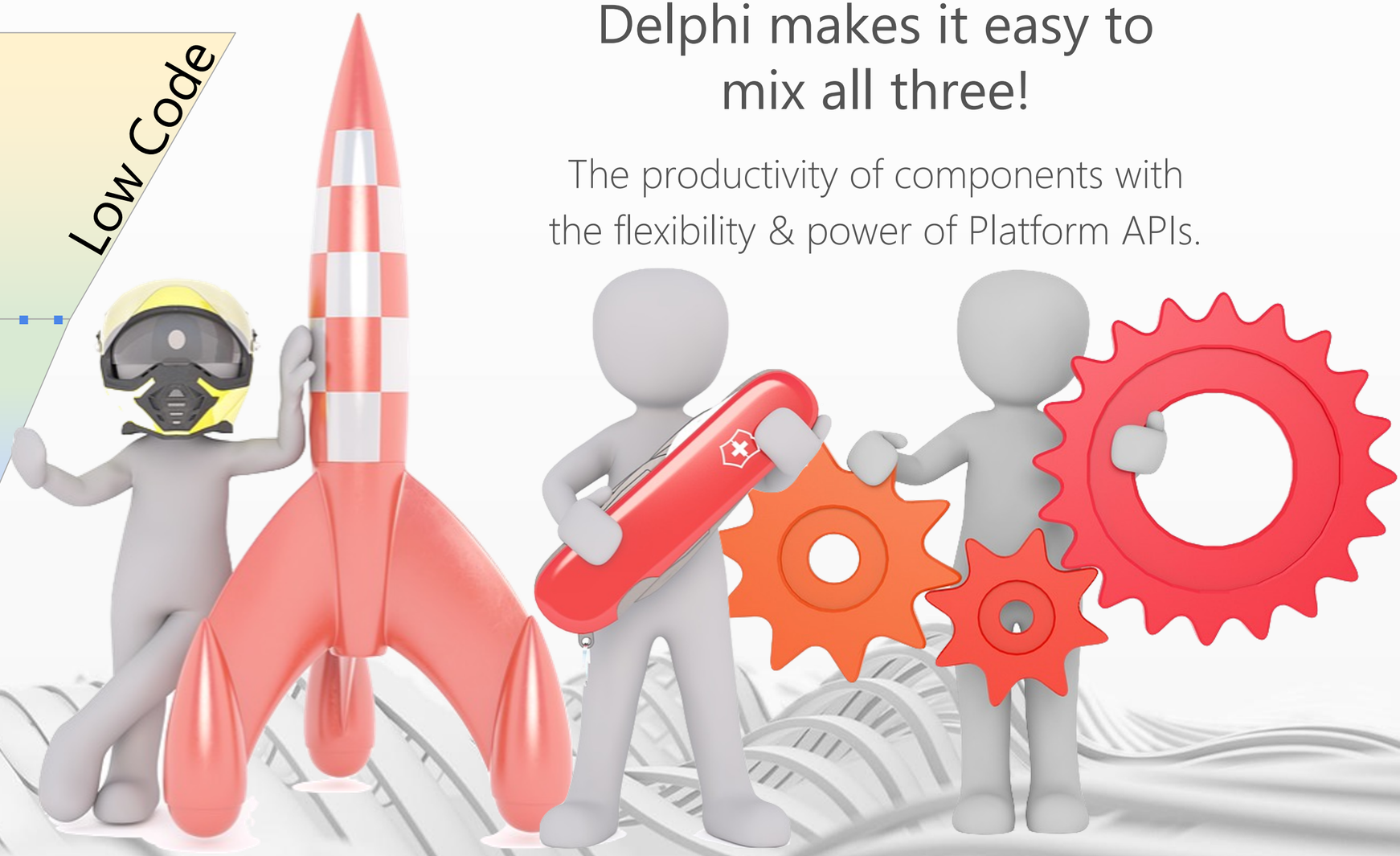
Raw Memory/ Assembly/ Pointers/ Platform APIs

Productivity

Where you "Touch the metal" for max flexibility

Delphi makes it easy to mix all three!

The productivity of components with the flexibility & power of Platform APIs.



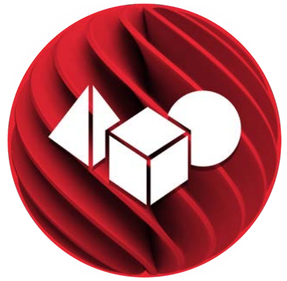
embarcadero®

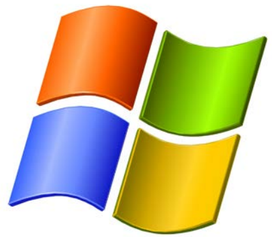
VCL vs FMX

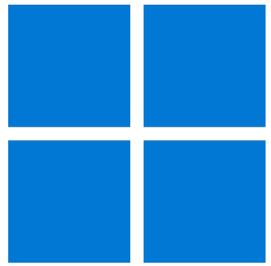
A tale of two GUI frameworks

VCL

Visual Component Library for Microsoft Windows







- Designed for Windows and evolving with each new Windows release supporting latest features, controls, themes, and design
- Full source ships with each release since Delphi 1
- Mostly a light wrapper around the native platform widgets and controls
- Includes a lot of “owner drawn” controls too
- Provides easy access to Windows Handles, Messages, etc.
- Greatly simplifies Windows development maintaining access to all platform features and APIs
- Native Windows look and feel with full theme system
- Mature platform with rich ecosystem of 3rd party components

FMX

The Cross-Platform FireMonkey Framework

- Takes advantage of GPU libraries to provide a hardware accelerated, rich user interface that is fast and looks great across multiple platforms:
  - Windows, macOS, iOS, Android, and Linux
  - Uses DirectX on Windows, OpenGL on Linux, OpenGL-ES on Android, and Metal on iOS and macOS
- Similar to VCL, but not designed to be compatible
  - Designed as cross platform from the ground up
- Integrated GPU effects, animations, and robust styling system
- Platform services abstract the access to platform hardware and functionality to intelligently adapt the UI & UX to platform specifics
- Very flexible component system - do more with fewer components (nestable)

Short Delphi Tour

Delphi IDE, VCL, & FMX

Short Delphi Tour

- Delphi isn't required to use the Delphi for Python modules
- The IDE runs on Windows, but targets other platforms
- There are multiple editions and options to get started
- Free options
  - 30-day trial of RAD Studio Enterprise
  - Community Edition (some limitations apply)
- Three paid editions
  - Professional
  - Enterprise
  - Architect
- *Note:* RAD Studio includes Delphi and C++Builder

Find out mote
embarcadero.com/products/delphi

Introduction to Delphi

embarcadero®

Getting Started with Delphi VCL for Python

Rich GUI Framework for Python

For Windows (32-bit & 64-bit)

embarcadero®

Delphi for Python

Delphi VCL for Python

- Windows 32-bit and 64-bit only
- Windows 8.1 through Windows 11
  - (Earlier versions may work, but not supported.)
- Based on native Windows components
- Includes Windows Handles, Messages, Accessibility, etc.
- Styling system

Delphi FMX for Python

Covering in webinar Part 2!

- Uses GPU for custom rendering
- Multi-platform for Windows, Linux, Android, and Mac OS
- Higher level of abstraction
- Platform services simplifies behaviors
- Styling system

Delphi VCL for Python Installation

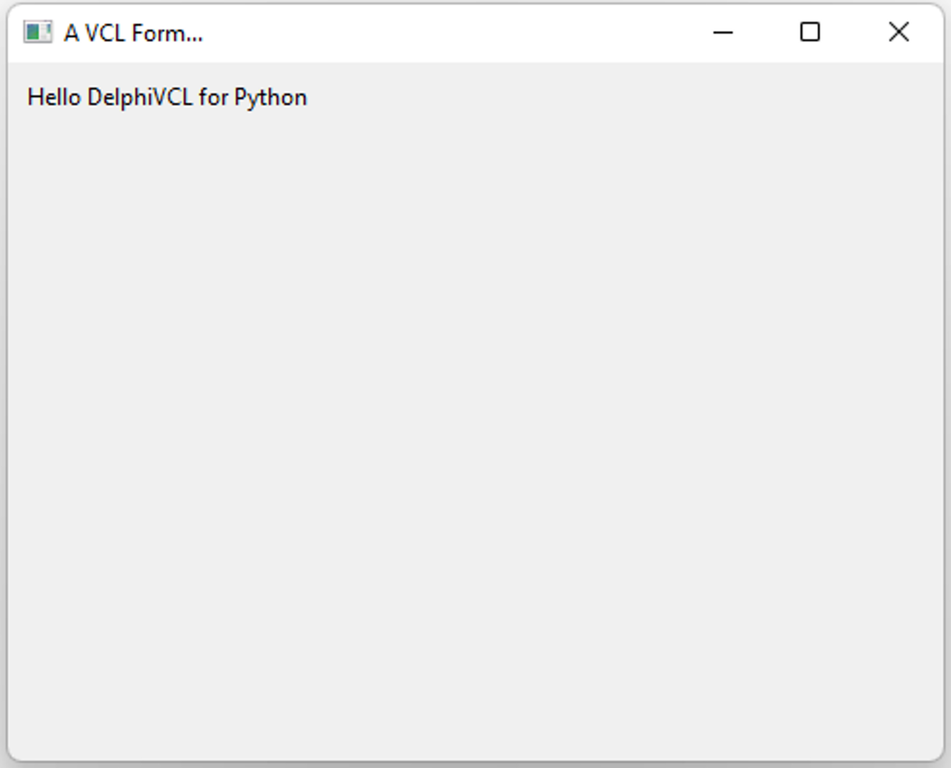
- Supports:
  - Win32 & Win64 x86 architectures
  - Python cp3.6, cp3.7, cp3.8, cp3.9 and cp3.10
- Conda support:
  - Win x86 and x64 from Python cp3.6 to cp3.10
- Install via pip
  - `pip install delphivcl`
- Details and downloads
  - github.com/Embarcadero/DelphiVCL4Python
  - pypi.org/project/delphivcl/

Hello Delphi VCL for Python

embarcadero®

Hello World

The simplest example




```
from delphivcl import *


class MainForm(Form):

    def __init__(self, owner):
        self.Caption = "A VCL Form..."
        self.SetBounds(10, 10, 500, 400)
        self.Position = "poScreenCenter"
        self.OnClose = self.__on_form_close

        self.lblHello = Label(self)
        self.lblHello.SetProps(Parent=self, \
            Caption="Hello DelphiVCL for Python")
        self.lblHello.SetBounds(10, 10, 300, 24)


    def __on_form_close(self, sender, action):
        action.Value = caFree


def main():
    Application.Initialize()
    Application.Title = "Hello Python"
    Main = MainForm(Application)
    Main.Show()
    FreeConsole()
    Application.Run()
    Main.Destroy()


main()
```

Class is a VCL Form

Owner

Configure the form

Create and configure the label

Event handler for the form's close event

Initialization the application and creating the form.

The program loop

Ownership
(Life-cycle)

- Application
  - Form1
    - Panel
    - Label
    - Edit
    - Button
    - Panel
    - Components
  - Form2
    - Components

Parent
(Drawing)

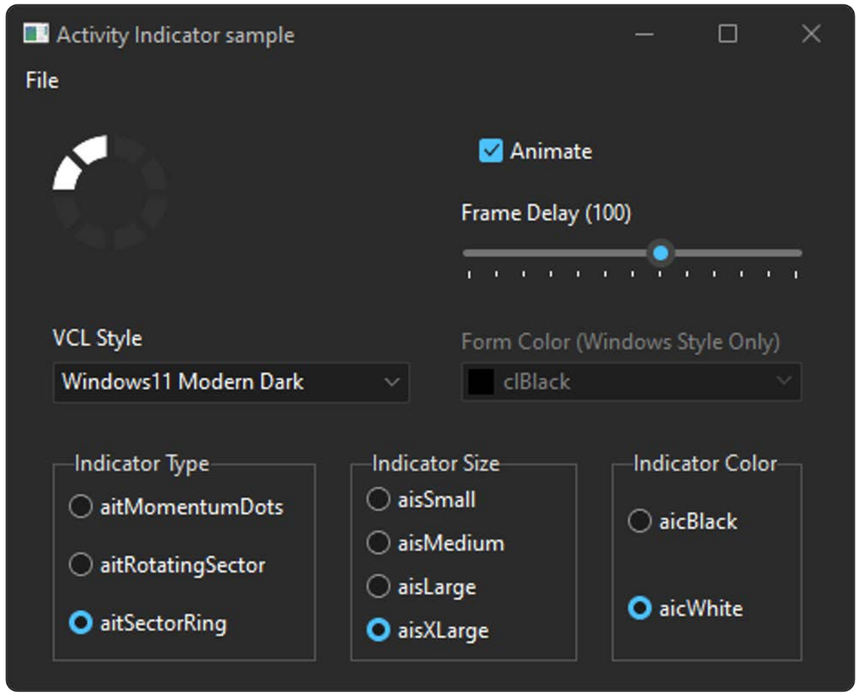
- Form1
  - Panel
    - Button
    - Panel
      - Label
      - Edit
- Form2
  - Nested Components
    - Nested Components
      - Nested Components

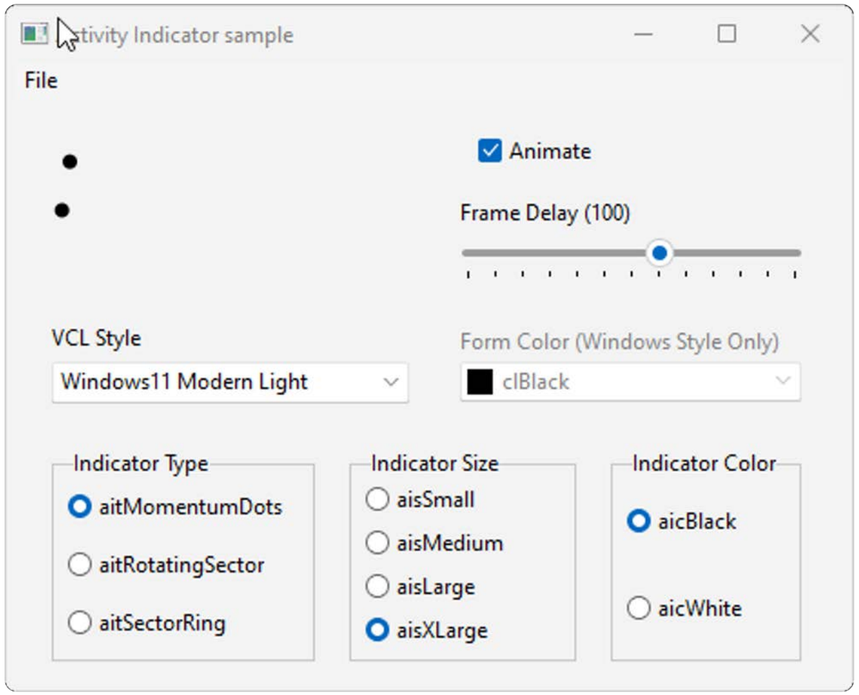
A component can not have an owner, but all visual components (except the Form) have a parent.

Parents *hold* their children, but don't necessarily *own* them.

Example with Styles

- Activity Indicator sample with option to load and change styles
- github.com/Embarcadero/DelphiVCL4Python/tree/main/samples/ActivityIndicator





Activity Indicator with Styles

embarcadero®

Going Further…

Exploring the
Delphi for Python
Bridge

embarcadero®

Design Your UI
in the Delphi IDE

- Take full advantage of the Delphi IDE designers and property editors
- WYSIWYG preview with styles
- Export the form for use in Python
- Requires no Object Pascal knowledge
- Still write all your code in Python
- Just right-click and export with the DelphiVCL4Python IDE add-in
- github.com/Embarcadero/DelphiVCL4Python/tree/main/experts

Form Export

embarcadero®

Combine Delphi and Python

- The Python4Delphi library is a bidirectional bridge
- Develop parts of your solution in Delphi, and part in Python
  - Play to the strengths of each
- Merge them together into a single cohesive solution
- Find samples, tutorials, and videos
  - github.com/pyscripter/python4delphi

Use Delphi to Create Native Python Modules

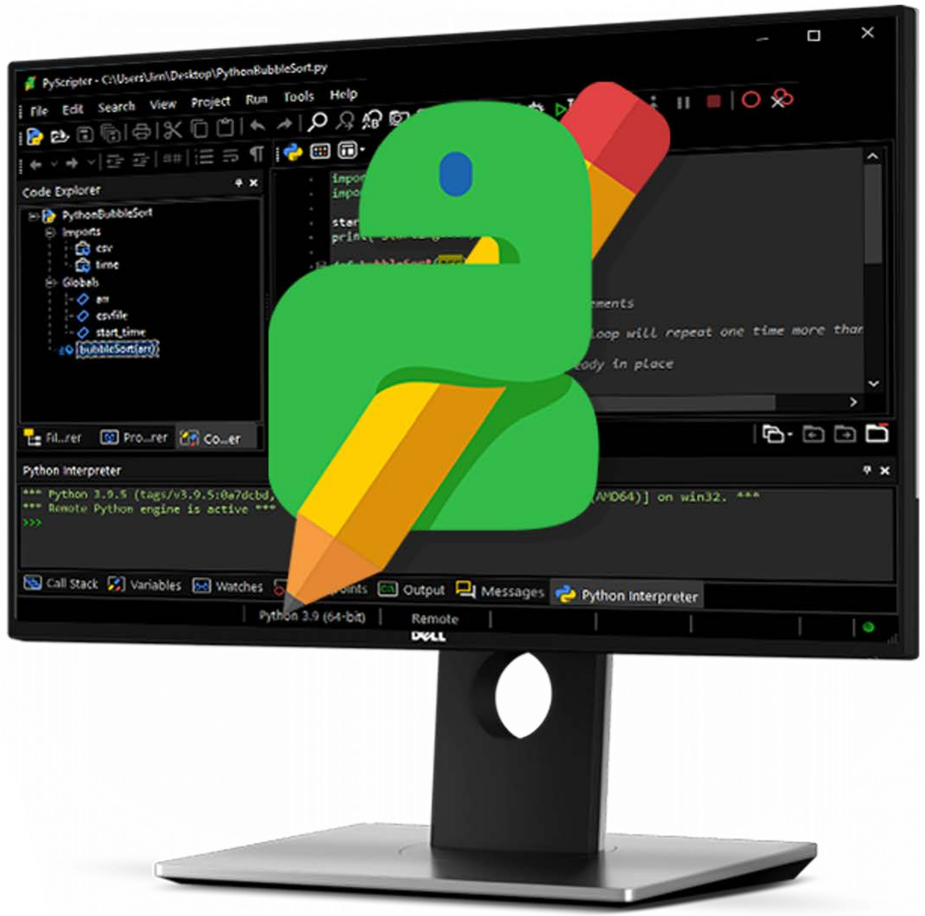
- Many Python modules are written in C/C++ and natively compiled
- Delphi also creates natively compiled Python modules via Python4Delphi
- Prototype rapidly in Python, and then create optimized modules in Delphi to clear bottlenecks
- Much like Python, Delphi code is focused on readability and clear structure and may be easier for you than using C/C++
- Augment your use of PyPy or Cython

Documentation and Library References

- Main Delphi documentation
  - Main docwiki.embarcadero.com/RADStudio/en/
  - VCL docwiki.embarcadero.com/RADStudio/en/VCL_Overview
  - FMX docwiki.embarcadero.com/RADStudio/en/FireMonkey
- Library Reference
  - Main docwiki.embarcadero.com/Libraries/en/
  - VCL docwiki.embarcadero.com/Libraries/en/Vcl
  - FMX docwiki.embarcadero.com/Libraries/en/FMX
- Delphi prefixes type names with a *"T"*
  - *TEdit* in Delphi is an **Edit** in Python
  - It is just a naming convention

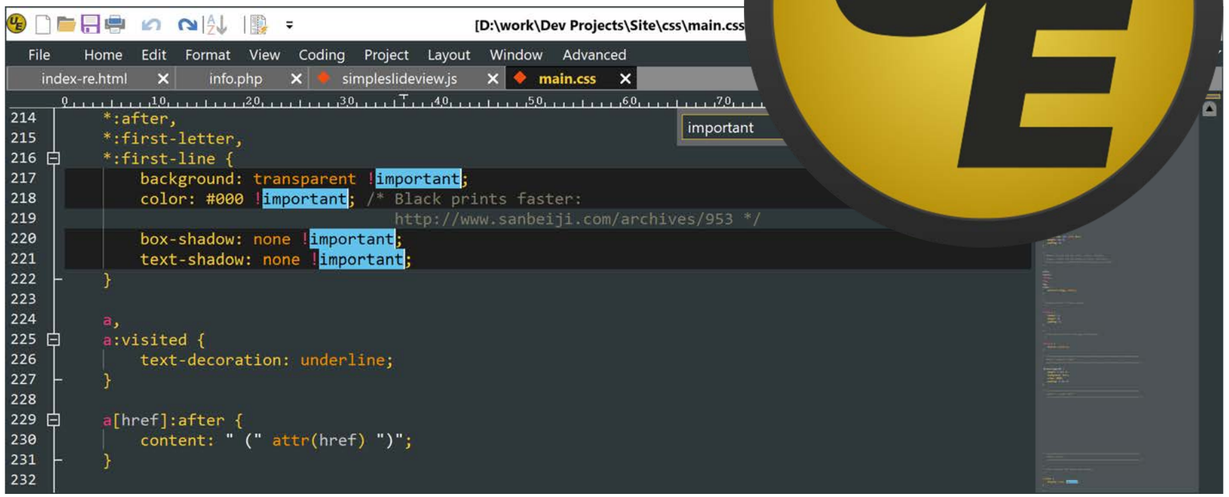
About PyScripter

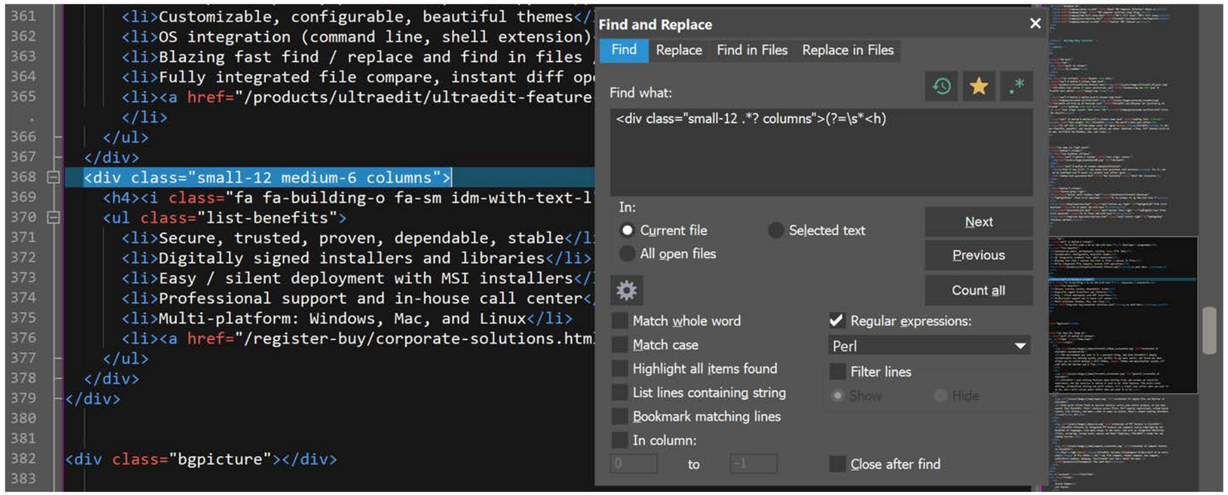
- Popular *free* & open-source Python IDE sponsored by Embarcadero
- All the features expected in a modern Python IDE while being lightweight and very fast
- Natively compiled for Windows to use minimal memory with maximum performance
- Full local and remote Python debugging
- Integration with Python tools like PyLint, TabNanny, Profile, etc.
- Run or debug files from memory.
- embarcadero.com/free-tools/pyscripter/free-download



About UltraEdit

- High performance text editor for programmers.
- Industry's best large file handling: 10+ GB and beyond.
- Syntax highlighting for nearly any language or data format.
- Smart templates.
- Hex editing. Column / block mode editing.
- Part of Idera family of developer tools.
- ultraedit.com/products/ultraedit/

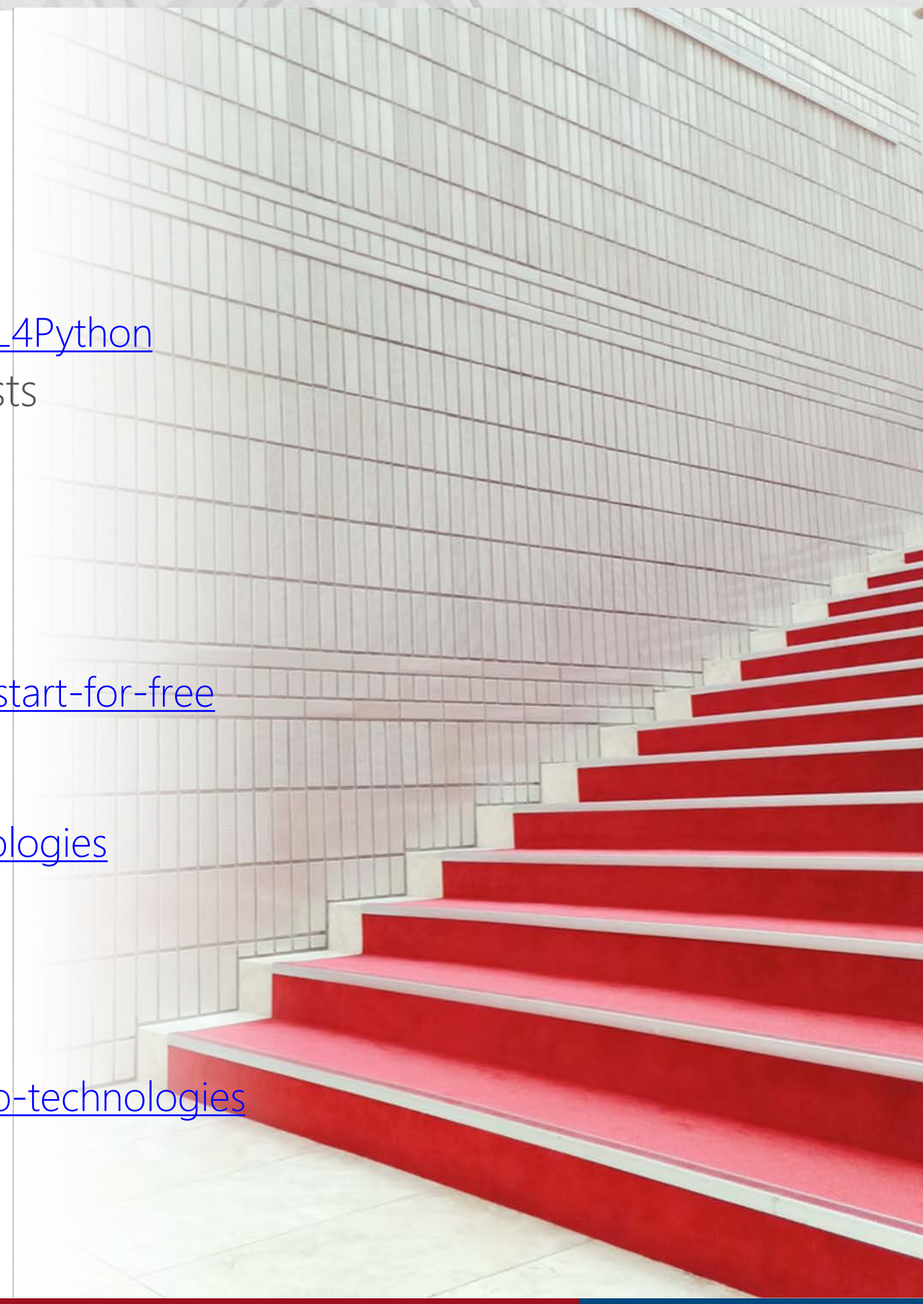




Next Steps

- Install Delphi VCL for Python — github.com/Embarcadero/DelphiVCL4Python
  - Star the repository, see the samples, file issues, and make feature requests
- Read the blog post (links, replays) — blogs.embarcadero.com/?p=128183
- Join part 2 on FMX & Android — https://embt.co/d4p-part2
  - Jan 26th at 9 AM CST *(Same registration as Part 1)*
- Start a 30-day Delphi trial — embarcadero.com/products/delphi/start-for-free

- Subscribe on YouTube — youtube.com/c/EmbarcaderoTechnologies
- Follow us on Twitter — twitter.com/embarcaderotech
- Like us on Facebook — facebook.com/embarcaderotech
- Follow us on LinkedIn — linkedin.com/company/embarcadero-technologies
- Read our blog — pythongui.org



Introduction to Python GUI Development with Delphi for Python

Part 1: Delphi VCL for Python

Q&A

Jim McKeeth
Chief Developer Advocate
Embarcadero Technologies
jim.mckeeth@embarcadero.com
@JimMcKeeth

embarcadero®

Slides, links & replay: blogs.embarcadero.com/?p=128183