



# **EDB Postgres™ Language Pack Guide**

**Version 11**

**October 18, 2018**

EDB Postgres™ Language Pack Guide, Version 11  
by EnterpriseDB® Corporation  
Copyright © 2018 EnterpriseDB Corporation. All rights reserved.

EnterpriseDB Corporation, 34 Crosby Drive Suite 100, Bedford, MA 01730, USA  
**T** +1 781 357 3390 **F** +1 978 589 5701 **E** [info@enterprisedb.com](mailto:info@enterprisedb.com) **www**.[enterprisedb.com](http://enterprisedb.com)

# Table of Contents

1	Introduction .....	4
1.1	Supported Platforms .....	4
1.2	Typographical Conventions Used in this Guide .....	5
2	Installing Language Pack .....	6
2.1	Invoking the Graphical Installer .....	6
2.2	Installing Language Pack with StackBuilder Plus .....	10
2.3	Configuring Language Pack on an Advanced Server Host .....	11
2.4	Configuring Language Pack on a PostgreSQL Host .....	12
3	Using the Procedural Languages .....	13
3.1	PL/Perl .....	14
3.2	PL/Python .....	15
3.3	PL/Tcl .....	16

# 1 Introduction

Language pack installers contain supported languages that may be used with EDB Postgres Advanced Server and EnterpriseDB PostgreSQL database installers. The languagepack installer allows you to install Perl, TCL/TK, and Python without installing supporting software from third party vendors. The Language Pack installer includes:

- TCL with TK version 8.6.8
- Perl version 5.26.2
- Python version 3.6.8

The Perl package contains the `cpan` package manager, and Python contains `pip` and `easy_install` package managers. There is no package manager for TCL/TK.

In previous Postgres releases, `plpython` was statically linked with ActiveState's python library. The Language Pack Installer dynamically links with our shared object for python. In ActiveState Linux installers for Python, there is no dynamic library. As a result of these changes, `plpython` will no longer work with ActiveState installers.

This document uses the term *Postgres* to mean either EDB Postgres Advanced Server or EDB PostgreSQL. For more information about using EDB Postgres products, please visit the EnterpriseDB website at:

<http://www.enterprisedb.com/documentation>

StackBuilder Plus is distributed with Advanced Server; Stack Builder (distributed with PostgreSQL) provides comparable functionality. This document uses the term *StackBuilder Plus* to mean either StackBuilder Plus or Stack Builder.

## 1.1 Supported Platforms

Language Pack v11 is tested on:

- EDB Postgres Advanced Server version 11
- PostgreSQL version 11

## 1.2 *Typographical Conventions Used in this Guide*

Certain typographical conventions are used in this manual to clarify the meaning and usage of various commands, statements, programs, examples, etc. This section provides a summary of these conventions.

In the following descriptions, a *term* refers to any word or group of words that are language keywords, user-supplied values, literals, etc. A term's exact meaning depends upon the context in which it is used.

- *Italic font* introduces a new term, typically in the sentence that defines it for the first time.
- Fixed-width (mono-spaced) font is used for terms that must be given literally such as SQL commands, specific table and column names used in the examples, programming language keywords, etc. For example, `SELECT * FROM emp;`
- *Italic fixed-width font* is used for terms for which the user must substitute values in actual usage. For example, `DELETE FROM table_name;`
- A vertical pipe | denotes a choice between the terms on either side of the pipe. A vertical pipe is used to separate two or more alternative terms within square brackets (optional choices) or braces (one mandatory choice).
- Square brackets [ ] denote that one or none of the enclosed terms may be substituted. For example, [ a | b ] means choose one of “a” or “b” or neither of the two.
- Braces { } denote that exactly one of the enclosed alternatives must be specified. For example, { a | b } means exactly one of “a” or “b” must be specified.
- Ellipses ... denote that the preceding term may be repeated. For example, [ a | b ] ... means that you may have the sequence, “b a a b a”.

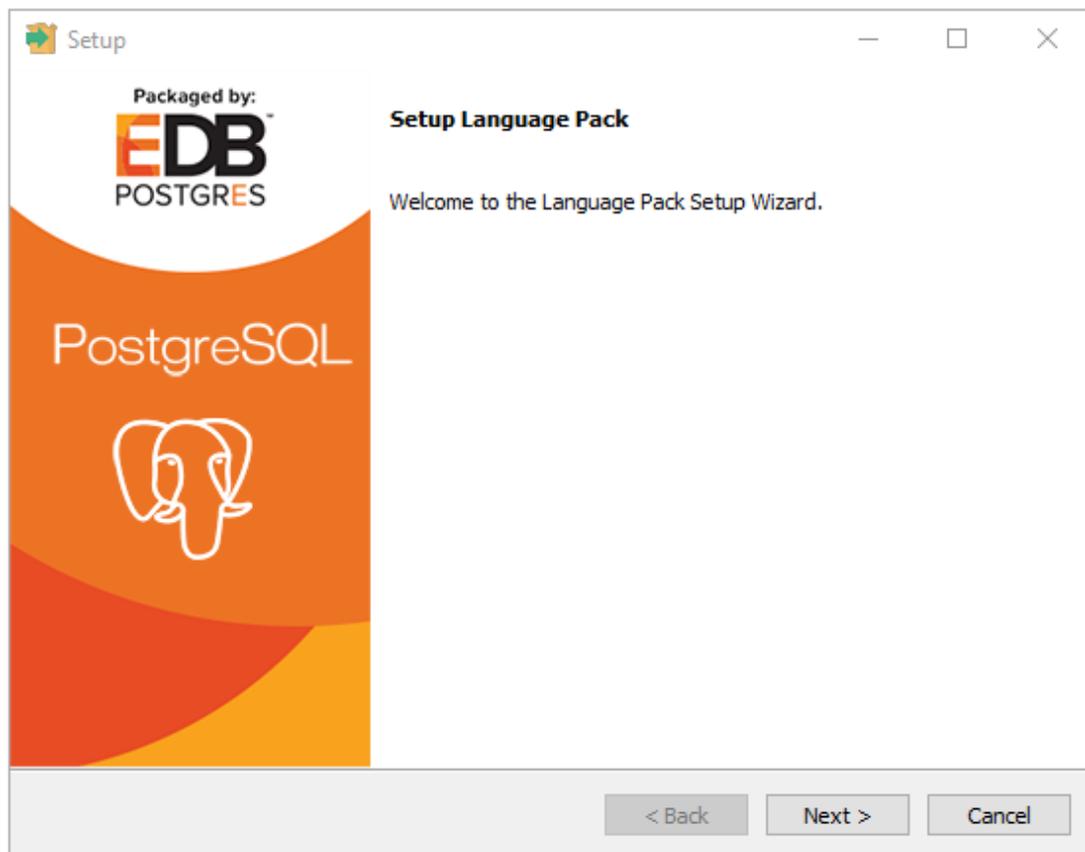
## 2 Installing Language Pack

The graphical installer is available from the EnterpriseDB website or via StackBuilder Plus.

### 2.1 Invoking the Graphical Installer

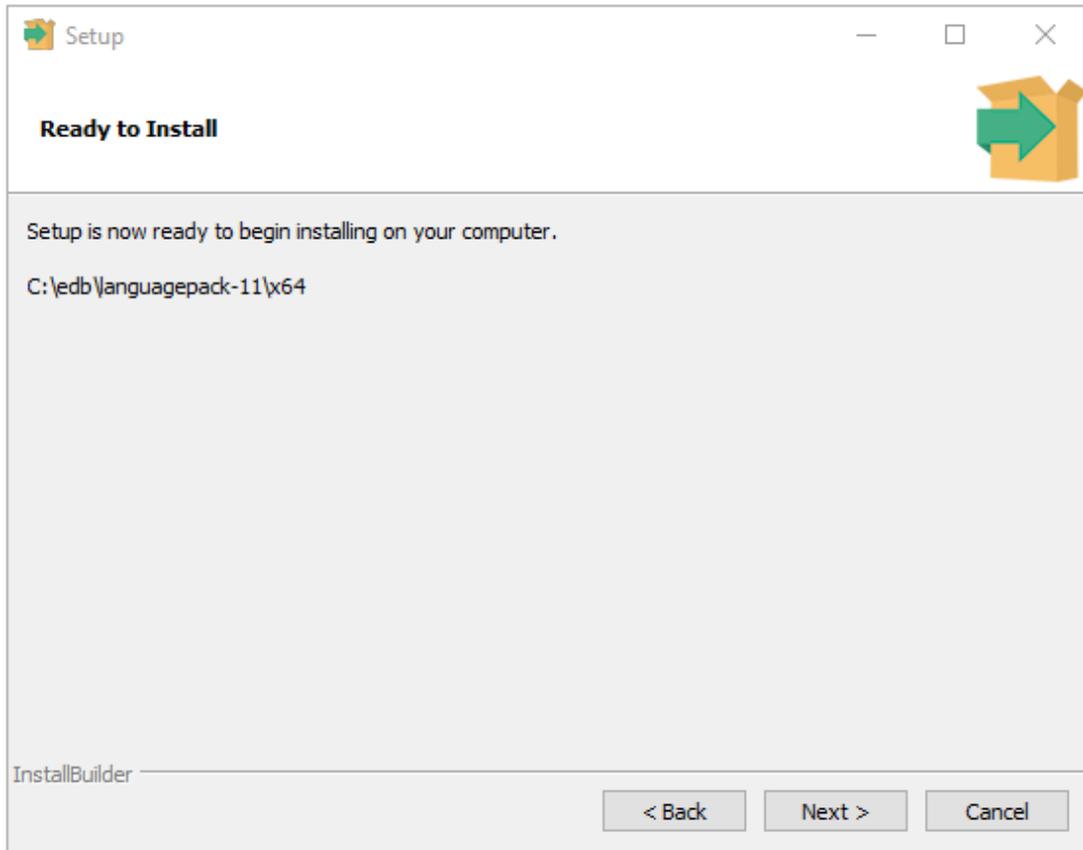
Assume `Administrator` privileges, and double-click the installer icon; if prompted, provide the password associated with the `Administrator` account. When prompted, select an installation language, and click `OK`.

The installer `Welcome` window opens (see Figure 2.1).



*Figure 2.1 – The Language Pack Welcome window.*

Click `Next` to continue.



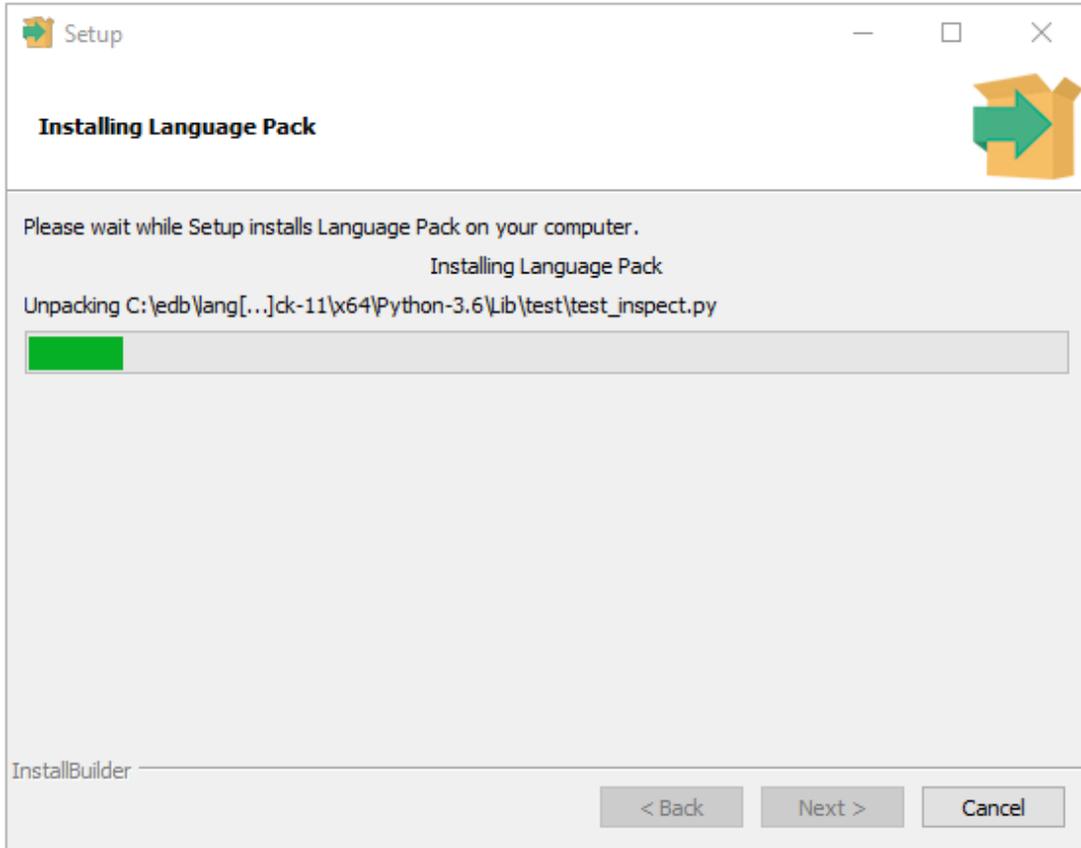
*Figure 2.2 – The Language Pack Welcome window.*

The `Ready to Install` window (see Figure 2.2) displays the Language Pack installation directory:

**On Windows 64:** `C:\edb\languagepack-11\x64`

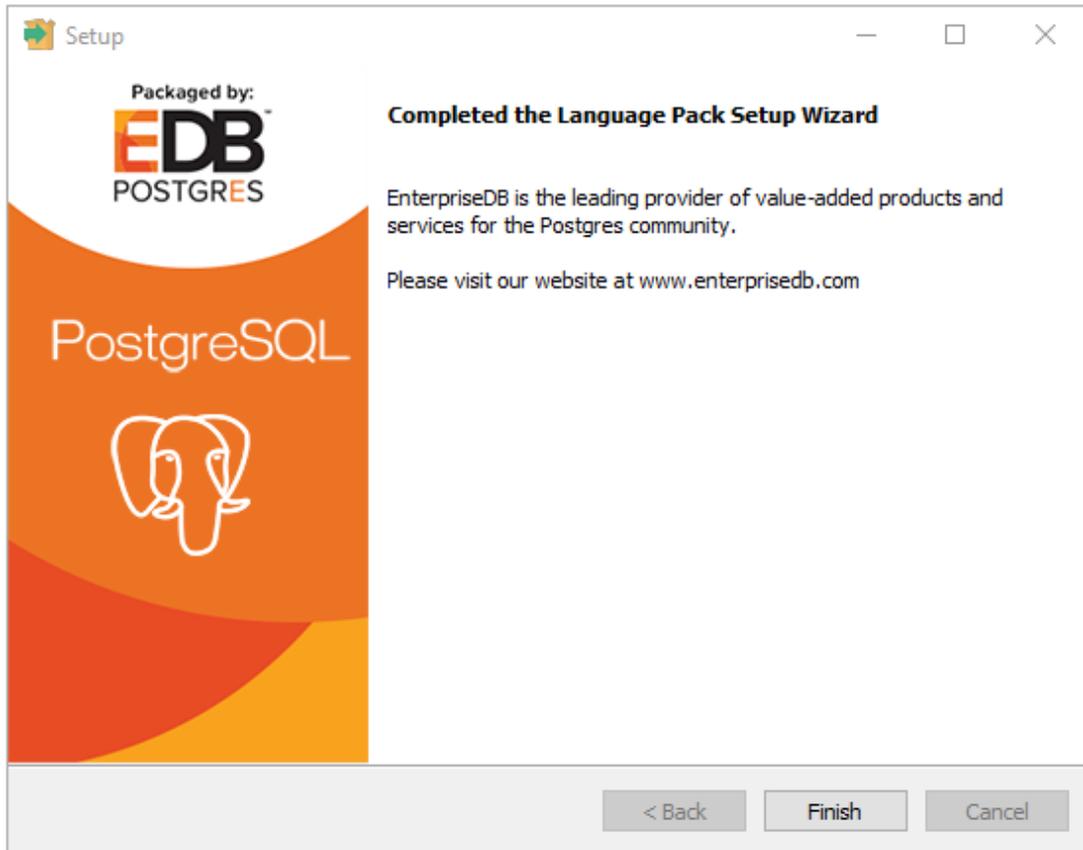
**On OSX:** `/Library/edb/languagepack-11`

You cannot modify the installation directory. Click `Next` to continue.



*Figure 2.3 – The Language Pack Welcome window.*

A progress bar marks installation progress (see Figure 2.3); click `Next` to continue.

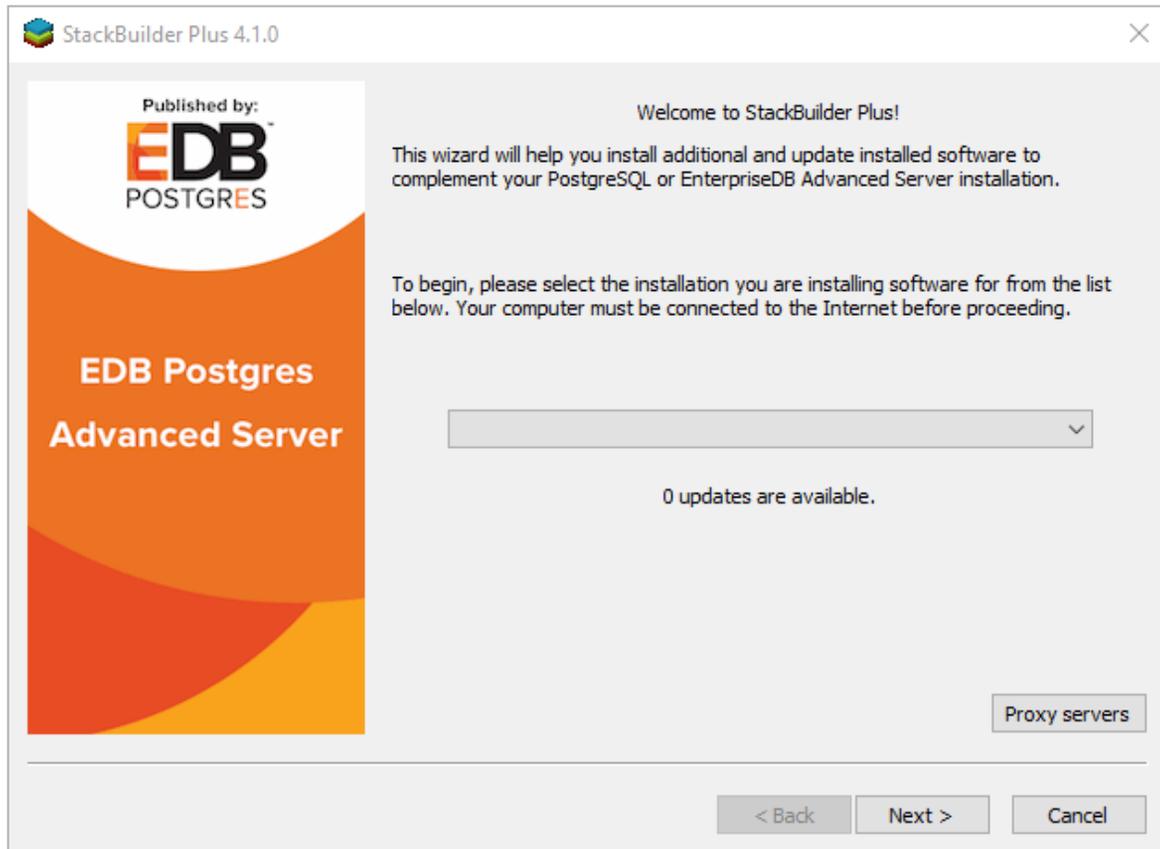


*Figure 2.4 – The Language Pack Welcome window.*

The installer will inform you that the Language Pack installation has completed (see Figure 2.4); click `Finish` to exit the installer.

## 2.2 Installing Language Pack with StackBuilder Plus

You can use StackBuilder Plus to download and invoke the Language Pack graphical installer. To open StackBuilder Plus, select the `StackBuilder Plus` menu item from the version-specific EDB Postgres sub-menu.



*Figure 2.5 – The StackBuilder Plus Welcome window.*

Select your server from the drop-down menu on the StackBuilder Plus `Welcome` window (see Figure 2.5) and click `Next` to continue.

Expand the `Add-ons, tools and utilities` node of the `Categories tree control`, and check the box to the left of `EDB Language Pack`; click `Next` to continue.

StackBuilder Plus will confirm your package selection before downloading the installer. When the download completes, StackBuilder Plus will offer to invoke the installer for you, or will delay the installation until a more convenient time.

For details about using the graphical installer, see Section [2.1](#).

## 2.3 *Configuring Language Pack on an Advanced Server Host*

After installing Language Pack on an Advanced Server host, you must configure the installation.

### *Configuring Language Pack on Windows*

On Windows, the Language Pack installer places the languages in:

```
C:\edb\languagepack-11\x64
```

After installing Language Pack, you must set the following variables:

```
set PYTHONHOME=C:\edb\languagepack-11\x64\Python-3.6.
```

Use the following commands to add Python, Perl and Tcl to your search path:

```
set PATH= C:\edb\LanguagePack-11\x64\Python-3.6;  
C:\edb\LanguagePack-11\x64\Perl-5.26\bin;  
C:\edb\LanguagePack-11\x64\Tcl-8.6\bin;%PATH%
```

After performing the steps required to configure Language Pack on Windows, use the Windows `Services` applet to restart the Advanced Server database server.

## 2.4 Configuring Language Pack on a PostgreSQL Host

After installing Language Pack on a PostgreSQL host, you must

### Configuring Language Pack on Windows

After installing Language Pack, you must tell the Python interpreter where to find Python:

```
set PYTHONHOME=C:\edb\languagepack-11\x64\Python-3.6
```

Then, use the following commands to add Language Pack to your search path:

```
set PATH= C:\edb\LanguagePack-11\x64\Python-3.6;
C:\edb\LanguagePack-11\x64\Perl-5.26\bin;
C:\edb\LanguagePack-11\x64\Tcl-8.6\bin;%PATH%
```

After setting the system-specific steps required to configure Language Pack on Windows, restart the database server.

### Configuring Language Pack on OSX

To simplify setting the value of `PATH` or `LD_LIBRARY_PATH`, you can create environment variables that identify the installation location:

```
PERLHOME=/Library/edb/languagepack-11/Perl-5.26
PYTHONHOME=/Library/edb/languagepack-11/Python-3.6
TCLHOME=/Library/edb/languagepack-11/Tcl-8.6
```

Then, instruct the Python interpreter where to find Python:

```
export PYTHONHOME
```

You can use the same environment variables when setting the value of `PATH`:

```
export PATH=$PYTHONHOME/bin:
$PERLHOME/bin:
$TCLHOME/bin:$PATH
```

Lastly, use the variables to tell OSX where to find the shared libraries:

```
export DYLD_LIBRARY_PATH=$PYTHONHOME/lib:
$PERLHOME/lib/CORE:$TCLHOME/lib:
$DYLD_LIBRARY_PATH
```

## 3 Using the Procedural Languages

The Postgres procedural languages (PL/Perl, PL/Python, and PL/Java) are installed with by the Language Pack installer. You can also use an RPM package to add procedural language functionality to your Advanced Server installation. For more information about using an RPM package, please see the EDB Advanced Server Installation Guide, available at:

<https://www.enterprisedb.com/resources/product-documentation>

### 3.1 PL/Perl

The PL/Perl procedural language allows you to use Perl functions in Postgres applications.

You must install PL/Perl in each database (or in a template database) before creating a PL/Perl function. Use the `CREATE LANGUAGE` command at the EDB-PSQL command line to install PL/Perl. Open the EDB-PSQL client, establish a connection to the database in which you wish to install PL/Perl, and enter the command:

```
CREATE EXTENSION plperl;
```

You can now use a Postgres client application to access the features of the PL/Perl language. The following PL/Perl example creates a function named `perl_max` that returns the larger of two integer values:

```
CREATE OR REPLACE FUNCTION perl_max (integer, integer) RETURNS integer AS
$$
    if ($_[0] > $_[1])
    { return $_[0]; }
    return $_[1];
$$ LANGUAGE plperl;
```

Pass two values when calling the function:

```
SELECT perl_max(1, 2);
```

The server returns:

```
perl_max
-----
         2
(1 row)
```

For more information about using the Perl procedural language, consult the official Postgres documentation available at:

<https://www.postgresql.org/docs/11/static/plperl.html>

## 3.2 PL/Python

The PL/Python procedural language allows you to create and execute functions written in Python within Postgres applications. The version of PL/Python used by Advanced Server and PostgreSQL is `plpython3u`; it offers no restrictions on users to prevent potential security risks.

Install PL/Python in each database (or in a template database) before creating a PL/Python function. You can use the `CREATE LANGUAGE` command at the EDB-PSQL command line to install PL/Python. Use EDB-PSQL to connect to the database in which you wish to install PL/Python, and enter the command:

```
CREATE EXTENSION plpython3u;
```

After installing PL/Python in your database, you can use the features of the PL/Python language.

Please note: The indentation shown in the following example must be included as you enter the sample function in EDB-PSQL. The following PL/Python example creates a function named `pymax` that returns the larger of two integer values:

```
CREATE OR REPLACE FUNCTION pymax (a integer, b integer) RETURNS integer
AS
$$
    if a > b:
        return a
    return b
$$ LANGUAGE plpython3u;
```

When calling the `pymax` function, pass two values as shown below:

```
SELECT pymax(12, 3);
```

The server returns:

```
pymax
-----
    12
(1 row)
```

For more information about using the Python procedural language, consult the official PostgreSQL documentation available at:

<https://www.postgresql.org/docs/11/static/plpython.html>

### 3.3 PL/Tcl

The PL/Tcl procedural language allows you to use Tcl/Tk functions in applications.

You must install PL/Tcl in each database (or in a template database) before creating a PL/Tcl function. Use the `CREATE LANGUAGE` command at the EDB-PSQL command line to install PL/Tcl. Use the `psql` client to connect to the database in which you wish to install PL/Tcl, and enter the command:

```
CREATE EXTENSION pltcl;
```

After creating the `pltcl` language, you can use the features of the PL/Tcl language from within your Postgres server.

The following PL/Tcl example creates a function named `tcl_max` that returns the larger of two integer values:

```
CREATE OR REPLACE FUNCTION tcl_max(integer, integer) RETURNS integer AS
$$
    if {[argisnull 1]} {
        if {[argisnull 2]} { return_null }
        return $2
    }
    if {[argisnull 2]} { return $1 }
    if {$1 > $2} {return $1}
    return $2
$$ LANGUAGE pltcl;
```

Pass two values when calling the function:

```
SELECT tcl_max(1, 2);
```

The server returns:

```
tcl_max
-----
      2
(1 row)
```

For more information about using the Tcl procedural language, consult the official Postgres documentation available at:

<https://www.postgresql.org/docs/11/static/pltcl.html>