PythonConnect

Documentation

Python Connect

Developers: Renan LECHIEN (ITAO, Belgium)
Jonas FERON (UCLouvain, Belgium)

Opensource: https://github.com/JonasFeron/PythonConnect

What is PythonConnect ?

- **Seamless Python Integration in C#** – Run Python 3 (latest version) directly within any C# application, combining the strengths of both languages.
- **Optimized Script Execution and Data Exchange** – Effortlessly run Python scripts, transfer data between languages, and leverage Python libraries (e.g., NumPy) within C# applications.
- **A Practical Alternative to Python.NET and IronPython** – Provides a different approach for integrating Python with C#, offering flexibility and ease of use.
- **Asynchronous and Parallel Execution** – Efficiently manages communication between C# and Python, running processes simultaneously without blocking workflows.
- **Conclusion:** PythonConnect simplifies sending data from C# to Python 3 and retrieving results, making cross-language integration smooth and efficient.

How to run python scripts from C# applications ?

State-of-the-art

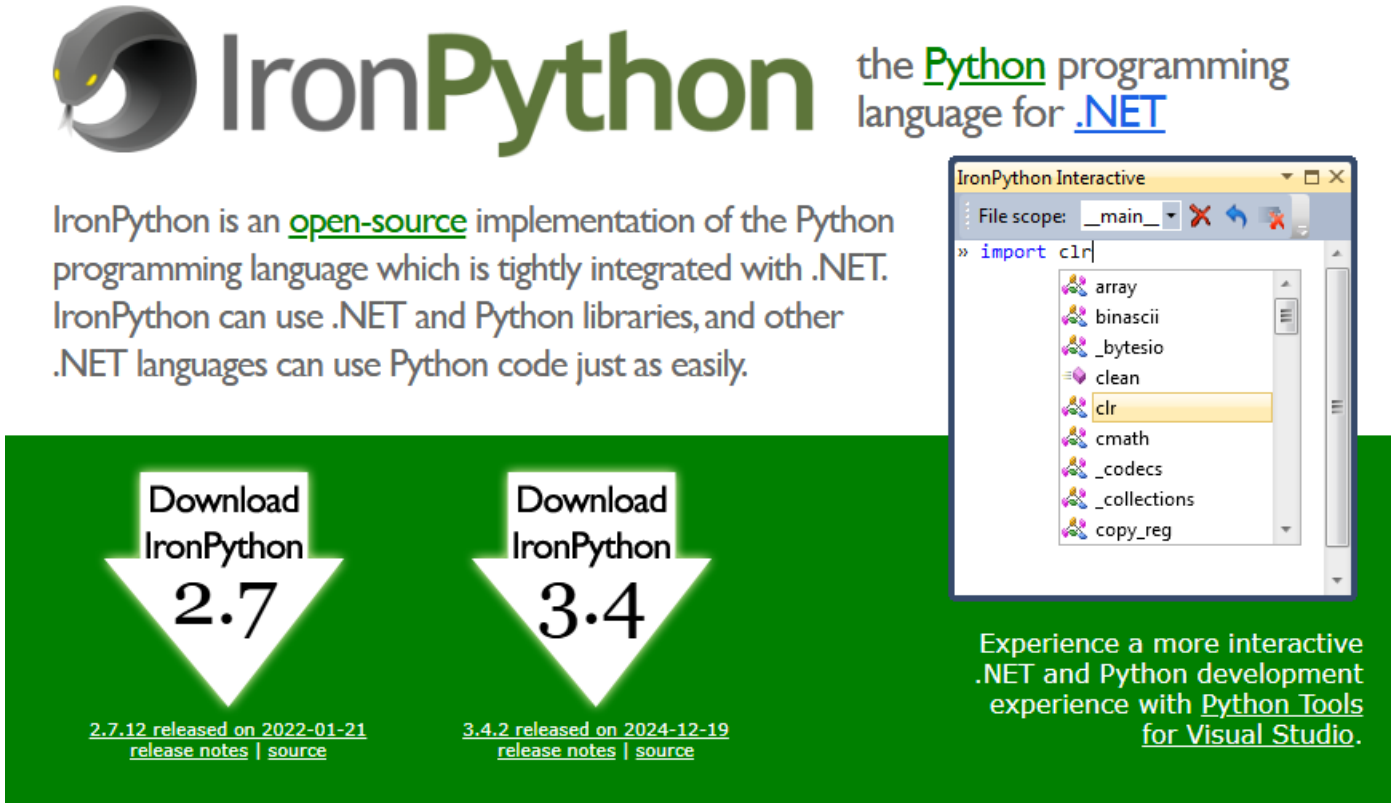
date: 2025.01.24

Existing method 1) IronPython

- IronPython 3.4 is based on Python3.4
- Limited to IronPython 2.7 up to April 19, 2021

> **April 19, 2021**
> IronPython 3.4.0-alpha1 is now available!
> See the release announcement for details.

- Today, last Python version is 3.13



Screenshots date : 2025.01.24

Source: IronPython.net

Existing method 2) Python.NET

Python.NET embeds Python into .NET:

- Import Python Librairies directly in C#
- Write C# code using Python Class and Methods

Example:

```
static void Main(string[] args)
{
        PythonEngine.Initialize();
        using (Py.GIL())
        {
                dynamic np = Py.Import("numpy");
                Console.WriteLine(np.cos(np.pi * 2));

                dynamic sin = np.sin;
                Console.WriteLine(sin(5));

                double c = (double)(np.cos(5) + sin(5));
                Console.WriteLine(c);

                dynamic a = np.array(new List<float> { 1, 2, 3 });
                Console.WriteLine(a.dtype);

                dynamic b = np.array(new List<float> { 6, 5, 4 }, dtype: np.int32);
                Console.WriteLine(b.dtype);

                Console.WriteLine(a * b);
                 Console.ReadKey();
        }
}
```

Source: Python.NET | pythonnet

Existing method 2) Python.NET

Conclusion: Python.NET is very powerful.

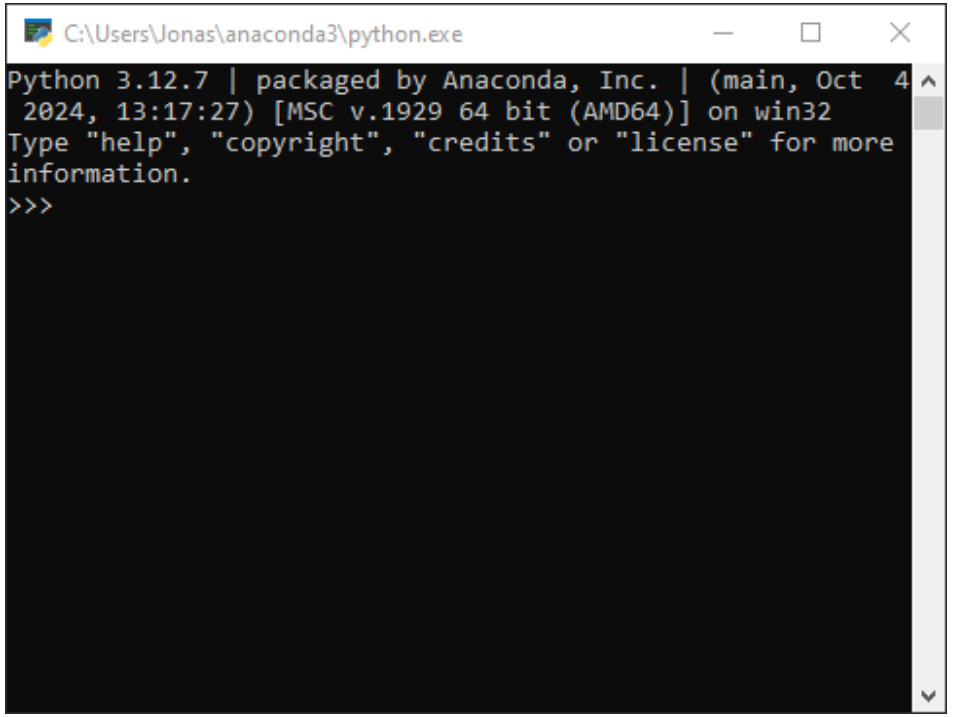
See pythonnet Wiki for more details.

To set up Python.NET for the first time, follow: Virtual Environments

For multi-threaded programs, follow: Multi-Threading

Existing method 3) Subprocess

Launch a Python "interpreteur" from C#



Example:

```
static void Main(string[] args)
{
        ProcessStartInfo start = new ProcessStartInfo();
        start.FileName = "python"; // Path to Python interpreter
        start.Arguments = « test_script.py"; // Path to Python script
        start.UseShellExecute = false;
        start.RedirectStandardOutput = true;

         using (Process process = Process.Start(start))
        {
                using (StreamReader reader = process.StandardOutput)
                {
                        string result = reader.ReadToEnd();
                        Console.WriteLine(result);
                }
        }
}
```

Source: ChatGPT 4o mini

PythonConnect

Introduction

Why PythonConnect?

Broad Python Compatibility – Use any Python version, including the latest release, without restrictions.

Seamless C# Integration – Run Python scripts directly in C# with no need for language translation.

Simple Setup – Easily manage Python environments and external libraries via Anaconda.org, minimizing dependency issues.

Multi-Threaded Execution – Send multiple Python commands simultaneously from C# for efficient handling of complex workflows.

Optimized Performance – Keep Python sessions active to avoid reinitialization overhead, significantly improving execution speed.

How PythonConnect works?

Prerequisite: Download and Install Anaconda

1. Parallel Thread Creation: Starting from the main C# thread, PythonConnect launches a Command Prompt subprocess in a second parallel thread.
2. Python Environment Initialization: The Command Prompt is then turned into an Anaconda environment capable of executing any Python 3 scripts.
3. Asynchronous Communication: Both the C# and Python threads operate simultaneously in an asynchronous manner. PythonConnect manages the communication between these threads, ensuring smooth data transfer from C# to Python 3 and retrieval of results back into C#.

Python Connect

PythonConnect

Explained

Simple example: python test_script.py

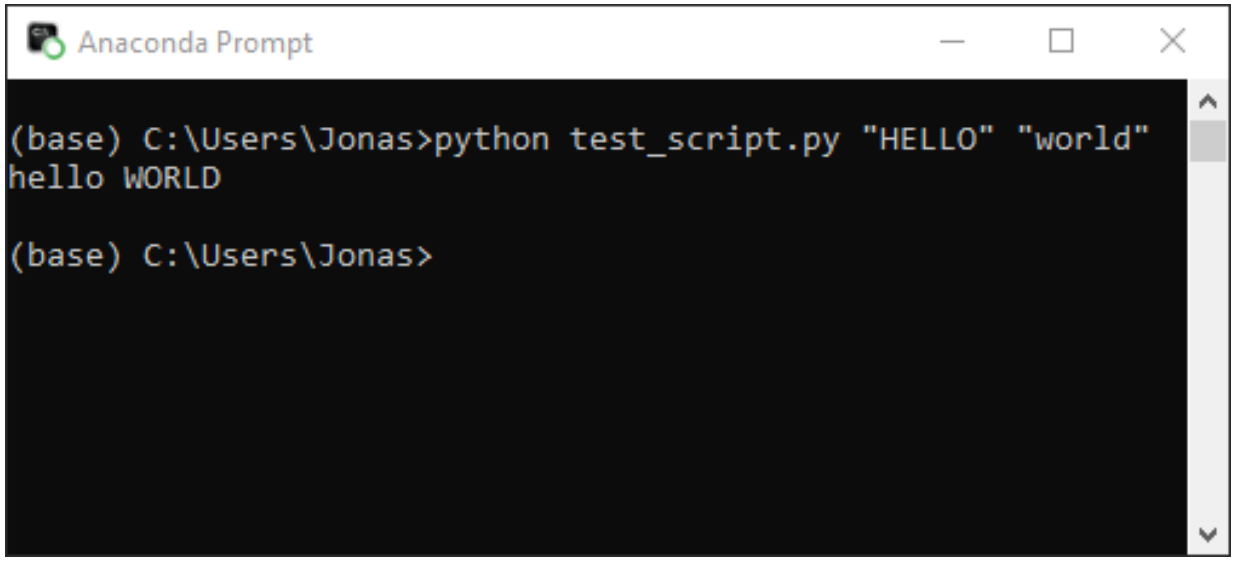
```python
# test_script.py
import sys

def main(arg1, arg2):
    return f"{arg1.lower()} {arg2.upper()}"


if __name__ == "__main__":
    arg1 = sys.argv[1]
    arg2 = sys.argv[2]
    result = main(arg1, arg2)
    print(result)
```

1. Function main(arg1, arg2)
   - Returns a string combining the lowercase of arg1 and uppercase of arg2.
2. if __name__ == "__main__":
   - Makes the script runnable.
   - Reads two arguments from the command line (sys.argv[1] and sys.argv[2]).
   - Prints the result of main(arg1, arg2).

PythonConnect:

behind the scene

Prerequisite:
Download and Install Anaconda




```python
# test_script.py
import sys

def main(arg1, arg2):
    return f"{arg1.lower()} {arg2.upper()}"


if __name__ == "__main__":
    arg1 = sys.argv[1]
    arg2 = sys.argv[2]
    result = main(arg1, arg2)
    print(result)
```

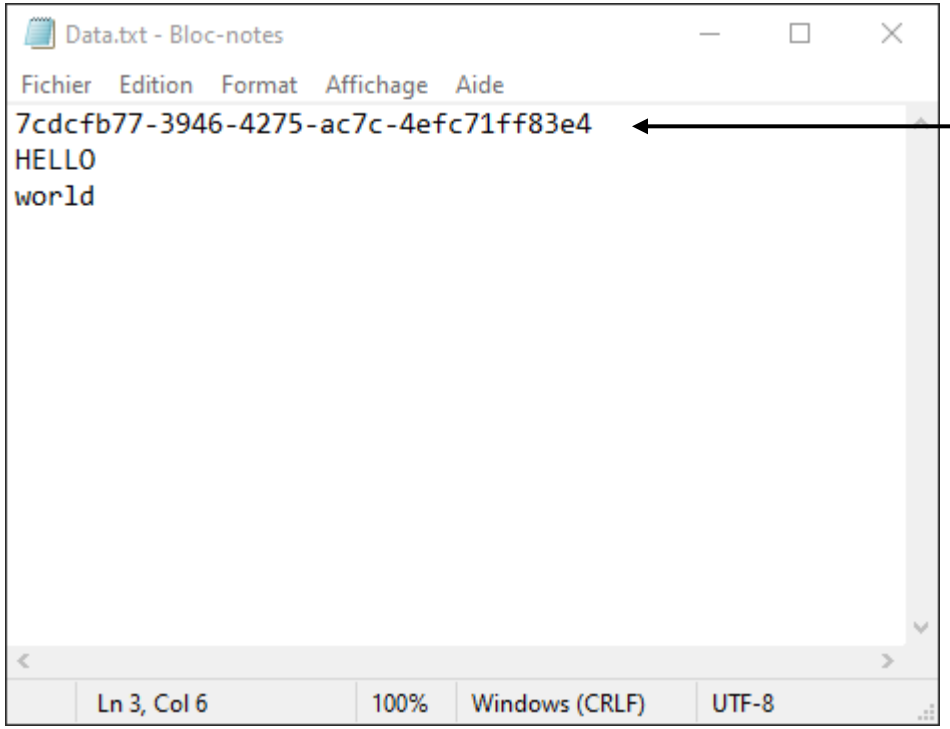
PythonConnect is based on existing method 3) subprocess.

Behind the scene, PythonConnect:

1. Launches from C# a subprocess “Anaconda Prompt”
2. Writes the command line to execute the script
3. Reads the result back in C#

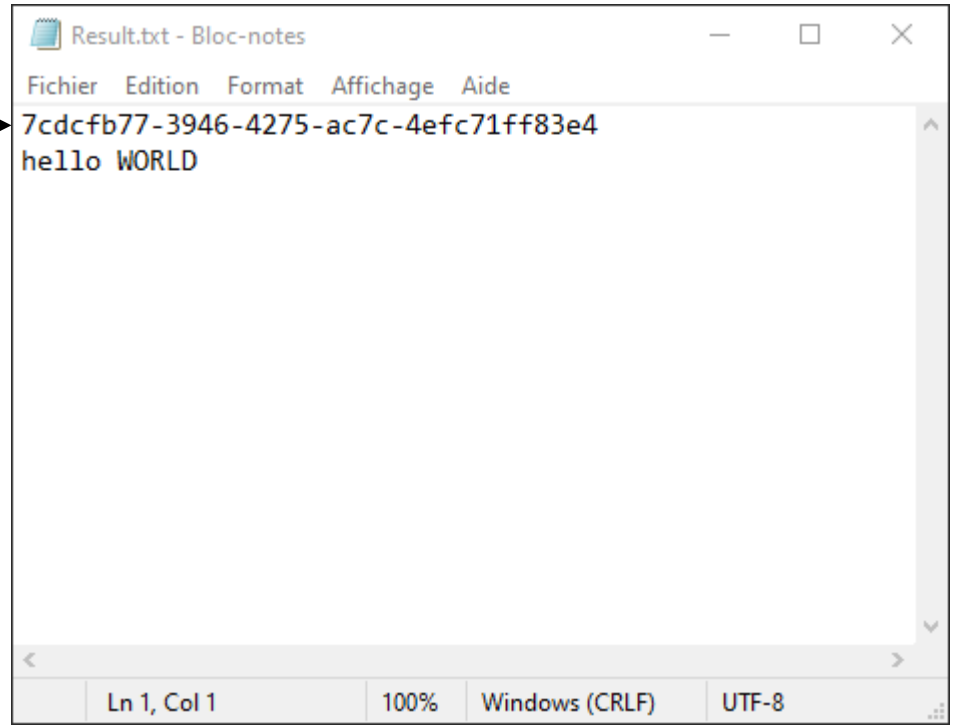
PythonConnect: behind the scene

Read/Write ‘Data.txt’ and ‘Result.txt’ files, for larger data exchanges



Unique ID

ensures that results always match data



ExecuteCommand from C#, in a single line

```
// Program.cs
using PythonConnect;

static void Main()
{
  string pythonScript = @"test_script.py";
  string dataPath = @“Path\To\Data.txt";
  string resultPath = @“Path\To\Result.txt";

  var result = pythonManager.ExecuteCommand(pythonScript, dataPath, resultPath, “HELLO", “world");

  Console.WriteLine($“{result}"); //C# output = “hello WORLD”
}
```

1. Main Program C# asks python to execute the python script `"test_script.py"` with arg1 = `“HELLO"` and arg2 = `“world“`
2. The arguments (arg1, arg2) are automatically sent to python through the file `“Data.txt"`
3. Results from python are automatically retrieved through the file `“Result.txt"`

Complex example

When arguments and results are complex data types

Data.txt - Bloc-notes

Fichier Edition Format Affichage Aide

```
73f16727-afbb-4e12-87d7-80f39ab02b21
{
"TypeName":"TwinData",

"List":[0.0,1.0,2.0,3.0,4.0,5.0,6.0,7.0,8.0,9.0],

"rowNumber":2,

"colNumber":5
}
```

Ln 5, Col 2 100% Windows

use JSON format

to automatically transform any data type into a string

Result.txt - Bloc-notes

Fichier Edition Format Affichage Aide

```
73f16727-afbb-4e12-87d7-80f39ab02b21
{
"TypeName":"TwinResult",

"Array":[[0.0,1.0,2.0,3.0,4.0],
         [5.0,6.0,7.0,8.0,9.0]]

}
```

Windows (CRLF) UTF-8

```python
# complex_script.py
import numpy as np

def main(List, rowNumber, colNumber):
    return np.array(List).reshape(rowNumber, colNumber)
```

PythonConnect is based on existing method 3) subprocess

But subprocess is single threaded ☹

In C#
- Write Data.txt
- Lauch new process: « Cmd»

In Cmd
- Activate Anaconda (SLOW)
- Run Python script

In Python 3
- Read Data.txt
- Do Stuff
- Write Result.txt

In Cmd
- Close Cmd

In C#
- Read Result.txt

Data - Notepad

File Edit Format View Help

```
38fe42e8-5962-4826-8601-152a729476d9
Hello
World
```

Invite de commandes

```
(c) Microsoft Corporation. Tous droits réservés.

C:\Users\Jonas>C:\Users\Jonas\anaconda3\Scripts\activate.bat

(base) C:\Users\Jonas>conda activate myenvironment

(myenvironment) C:\Users\Jonas>python test_script.py "Path\To\Data.txt"
```

Result - Notepad

File Edit Format View Help

```
38fe42e8-5962-4826-8601-152a729476d9
hello WORLD
```

Problems:

1. C# is blocked while the Command Prompt is open → Cmd must be closed after each execution.
2. Activating Anaconda is very slow ! And, it needs to be reactivated for each execution !

PythonConnect is based on existing method 3) subprocess

But PythonConnect is multi-threaded ☺

Python thread

In Cmd

- Activate Anaconda
  SLOW
  but only once !

Cmd is running behind the scene, waiting for new commands to execute.....

Invite de commandes

```
(c) Microsoft Corporation. Tous droits réservés.

C:\Users\Jonas>C:\Users\Jonas\anaconda3\Scripts\activate.bat

(base) C:\Users\Jonas>conda activate myenvironment

(myenvironment) C:\Users\Jonas>
```

C# threads

In C#

- Write Data.txt

New signal:
ExecuteCommand

Wait signal..... :
Ended execution

While other C# threads
do something else

In C#

- Read Result.txt

Conclusion: ExecuteCommand from C#, in just a single line

```
// Program.cs
using PythonConnect;

static void Main()
{
  string pythonScript = @"test_script.py";
  string dataPath = @“Path\To\Data.txt";
  string resultPath = @“Path\To\Result.txt";

  var result = pythonManager.ExecuteCommand(pythonScript, dataPath, resultPath, “HELLO", “world");

  Console.WriteLine($“{result}"); //C# output = “hello WORLD”
}
```

1. Main Program C# asks python to execute the python script `"test_script.py"` with arg1 = `“HELLO"` and arg2 = `“world“`
2. The arguments (arg1, arg2) are automatically sent to python through the file `“Data.txt"`
3. Results from python are automatically retrieved through the file `“Result.txt"`

Getting started with PythonConnect

1) Download and Install Anaconda

Getting started with PythonConnect

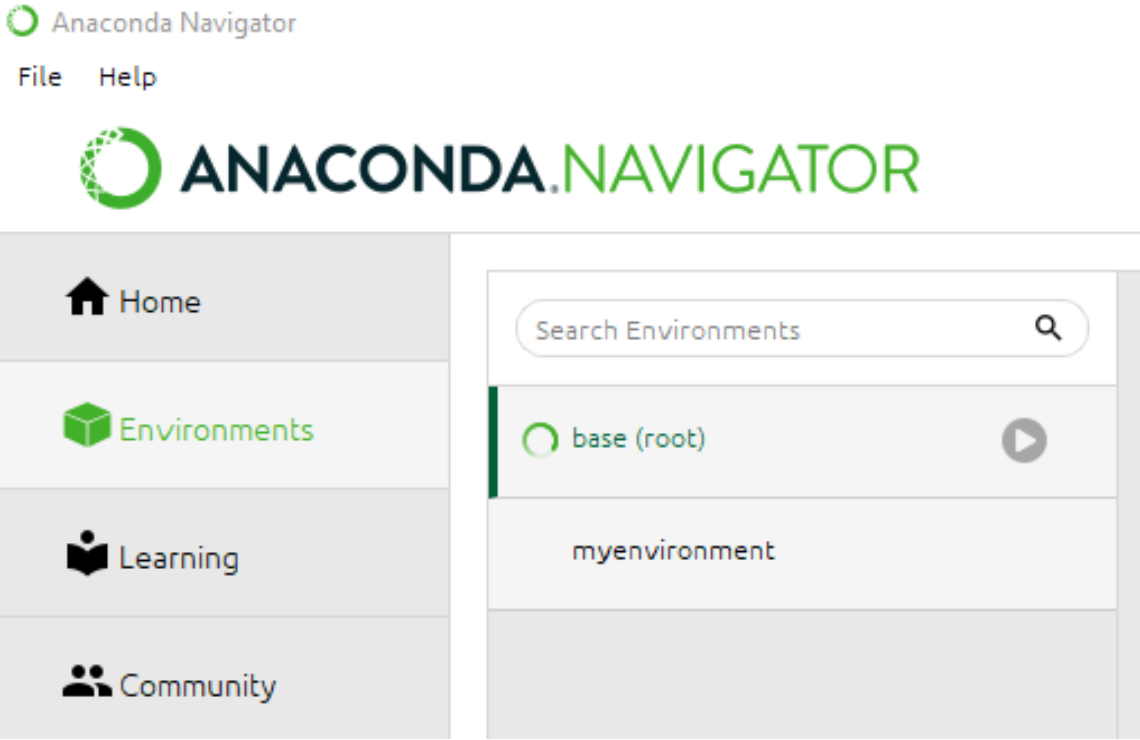
2) Manage python virtual environment

1. Use (base) conda environment
2. Or create a new environment for specific python version

via Command Prompt

```
Invite de commandes
(c) Microsoft Corporation. Tous droits réservés.

C:\Users\Jonas>C:\Users\Jonas\anaconda3\Scripts\activate.bat

(base) C:\Users\Jonas>conda create --name myenvironment python=3.12
Retrieving notices: ...working... done
Channels:
 - defaults
Platform: win-64
Collecting package metadata (repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: C:\Users\Jonas\anaconda3\envs\myenvironment

  added / updated specs:
    - python=3.12


The following packages will be downloaded:
```

Or via Anaconda Navigator App



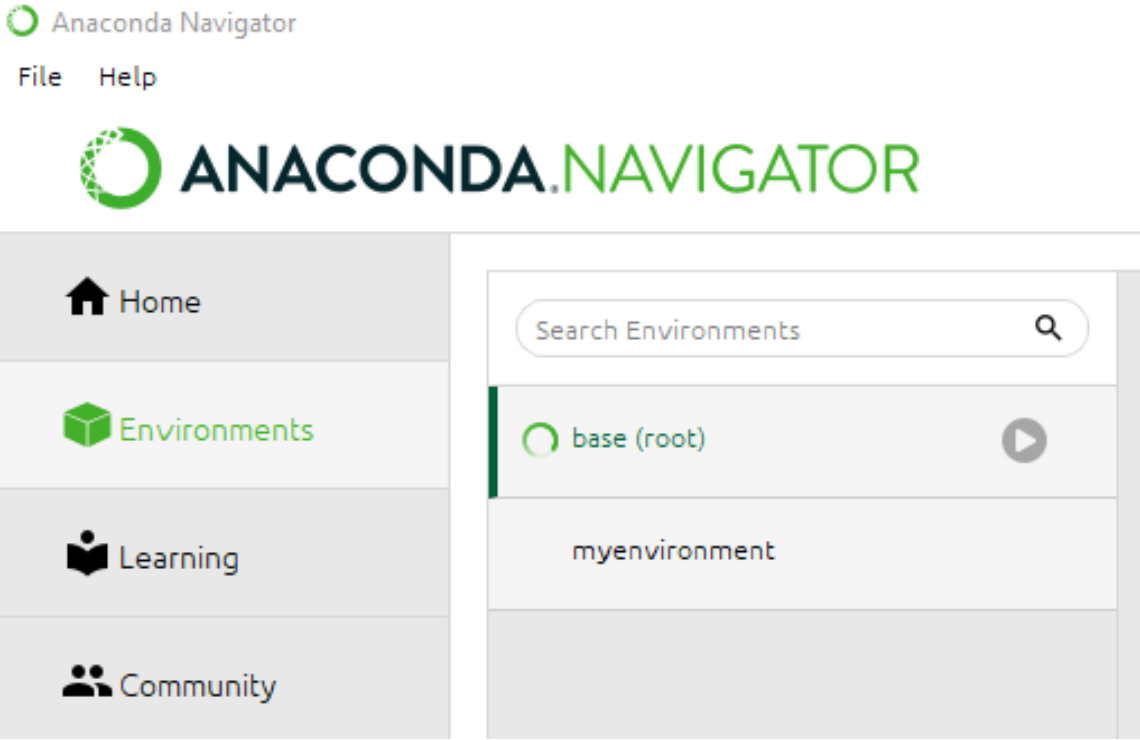
Getting started with PythonConnect

3) Install required python librairies in the environment

via Command Prompt

```
Invite de commandes
(c) Microsoft Corporation. Tous droits réservés.

C:\Users\Jonas>C:\Users\Jonas\anaconda3\Scripts\activate.bat

(base) C:\Users\Jonas>conda activate myenvironment

(myenvironment) C:\Users\Jonas>python --version
Python 3.12.8

(myenvironment) C:\Users\Jonas>pip install numpy
Collecting numpy
  Using cached numpy-2.2.2-cp312-cp312-win_amd64.whl.metadata (60 kB)

Using cached numpy-2.2.2-cp312-cp312-win_amd64.whl (12.6 MB)
Installing collected packages: numpy
Successfully installed numpy-2.2.2

(myenvironment) C:\Users\Jonas>
```

Or via Anaconda Navigator App



Getting started with PythonConnect

4) From https://github.com/JonasFeron/PythonConnect

- Clone the main branch of the Github repository, locally on your computer (GitHub Desktop helps)
- Open file PythonConnect.sln using Visual Studio
- In TestProject/Program.cs : Setup paths to your Anaconda installation
- Run TestProject/Program.cs

Getting started with PythonConnect

5) Create your own PythonConnected C# project

- Create a new C# solution and install the NuGet package PythonConnect
- From the GitHub repository, copy/paste the file src/python_connect/mainhelper.py in your solution
- Take insparation from the following repository for example with complex data exchange:

  https://github.com/JonasFeron/PythonConnectedGrasshopperTemplate