1. **Installation procedures**
2. Open the internet explorer / chrome browser in Windows 10 operating system
3. Type Scilab.org
4. Click on Download
5. Click on Scilab 6.1.0 - Windows 64 bits (exe) under the Windows Vista, 7, 8, 10 category
6. Check the downloads folder for the scilab-6.1.0\_x64.exe file.
7. If you find the file, then double click the file scilab-6.1.0\_x64.exe
8. Complete the Scilab installation after agreeing the terms and clicking the next buttons
9. Open the internet explorer / chrome browser in Windows 10 operating system again
10. Type the webpage <https://www.python.org/downloads/>
11. Click on Download Python 3.8.3
12. Follow the steps and click the next button and complete the installation of Python 3.8.3 on windows
13. Check the location of the installation by the following method



1. Click on the windows icon and type Python
2. Hover the mouse arrow over Python 3.8
3. Click on the open file location
4. In my computer the file location is “C:\Users\Subbulakshmi T\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Python 3.8”
5. Copy the file location by clicking on the file explorer address bar and pressing CTRL+C
6. Open command prompt and run as administrator
7. Type cd and right click.
8. The copied path from the file explorer must be available after the cd command
9. Click enter after right click
10. Check for the change in path
11. Now we are inside the Python 3.8.3 folder
12. Check by typing pwd command
13. Type pip install numpy
14. Download the SciPy-v0.1.9\_c.rar from fossee.in 🡪 hackathon
15. Copy the file to another location. I have copied to E:\softwares\Scilab
16. Unzip the file from the copied location
17. A new folder will be created with the name E:\softwares\Scilab\ SciPy-v0.1.9\_c
18. Click on the windows icon
19. Open Scilab 6.1.0 by typing Scilab
20. In the Scilab prompt type exec(‘E:\softwares\Scilab\ SciPy-v0.1.9\_c\loader.sce’)
21. Now the newly developed Python toolbox called as SciPy is loaded into Scilab for executing the Python calls inside the Scilab
22. All the following Python examples could be tried inside the Scilab environment now.

**Starting python interpreter**

**Function : startPy()**

Description : Starts the python interpreter, must be called before using any of the

other functions offered by SciPy.

**Function : quitPy()**

 Description : Stops the python interpreter, erasing all the information. Once called,

startPy must again be called before using any other functions offered by SciPy.

|  |  |  |  |
| --- | --- | --- | --- |
| **Sl.No** | **Parameters** | **Scilab example**  | **Python example** |
| 1. | Data type : integer  |
|  | Example  | a=10type(a) | py("x=1; y=type(x)")pyGet("y") |
|  | Output | ans = 1//1 corresponds to a real or complex matrix of double. | <class 'int'> |

**Loading the Built in functions**

Function : pyImport()

Syntax : <object\_name> = pyImport(“ <argument> “)

Description : Argument is the name of the module. One argument expected.

Example :

Default word to be used for all the built in functions in Python inside the Scilab is “builtins”. All other modules can be imported after external installation using pip install command as mentioned in the step 25.

py=pyImport("builtins")

This single command will load all the Python built in functions inside Scilab. Let us check using the table given below.

|  |  |  |  |
| --- | --- | --- | --- |
| **Sl.No** | **Parameters** | **Scilab Syntax and example**  | **Python Syntax and example** |
| 1. | Built in function name : abs()Description : Expects 1 argument. <value/variable> can be positive, negative or 0, returned value will be positive or 0.  |
|  | Syntax | <object\_name>.abs(<value/variable>) <object\_name> is to call builtins methods.  | abs(<value/variable>) |
|  | Example1  | -->var1=Py.abs(-9) | >> var2=abs(-5) |
|  | Output | var1=9 | >> var25 |
|  | Example 2 | -->Py.abs(9) | >>abs(-5) |
|  | Output  | ans = 9 | 5 |

np=pyImport("numpy")

(import numpy as np for python)

This single command will load all the numpy functions inside Scilab. Let us check using the table given below.

|  |  |  |  |
| --- | --- | --- | --- |
| **Sl.No** | **Parameters** | **Scilab Syntax and example**  | **Python Syntax and example** |
| 1. | Built in function name : array()Description : Expects 1 argument. <value/variable> can be a list of values or single value of any data type or any lists to make multidimensional array.  |
|  | Syntax | <object\_name>.array(pyList(pyList(value,value,…),pyList(value,value,….),….))<object\_name> is to call builtins methods.  | np.array([<list of values/list variable>],[<list of values/list variable>],…..)  |

**Executing Python code in Scilab**

Function : py()

Syntax : py(“<python\_code>”)

Description : Executes any simple python code passed and returns the console output.

Example : py("x=2; x=x+4; print(x)")

Output : ans = “6”

Function : pyExec()

Syntax : pyExec(“<path\_to\_python\_file>”)

Description : Executes any simple python code passed and returns the console output.

Example : pyExec("C:\\Users\\Dheekksha\\Downloads\\Scithon-v0.1.12\_c\\Scithon-v0.1.12\_c\\python\\Tools\\demo\\hi.py")

Output : ans = “hello world”

Function : pyGet()

Syntax : pyGet(“<python\_variable>”)

Description : Retrieves a named variable previously defined through py() or pyExec().

Example : py("x=2;") pyGet("x")

Output : ans = 2

Function : pyEquals()

Syntax : pyEquals(<argument1>,<argument2>)

Description : Compares the two arguments and returns the result; equivalent to arg1 == arg2. Arguments can be of type int, double, string, bool or python variable.

Example : x=2

 y=2

 pyEquals(x,y)

Output : ans = T

|  |  |  |  |
| --- | --- | --- | --- |
| **Prg.No** | **Parameters** | **Scilab Output** | **Python Output** |
| 1 | #Python code import sysn = 5if sys.argv[1:]: n = int(sys.argv[1])def bottle(n): if n == 0: return "no more bottles of beer" if n == 1: return "one bottle of beer" return str(n) + " bottles of beer"for i in range(n, 0, -1): print(bottle(i), "on the wall,") print(bottle(i) + ".") print("Take one down, pass it around,") print(bottle(i-1), "on the wall.") |
|  | Execution | pyExec("C:\\Users\\Dheekksha\\Downloads\\Scithon-v0.1.12\_c\\Scithon-v0.1.12\_c\\python\\Tools\\demo\\beer.py")  | Run module in IDLE |
|  | Output | ans = "5 bottles of beer on the wall,5 bottles of beer.Take one down, pass it around,4 bottles of beer on the wall.4 bottles of beer on the wall,4 bottles of beer.Take one down, pass it around,3 bottles of beer on the wall.3 bottles of beer on the wall,3 bottles of beer.Take one down, pass it around,2 bottles of beer on the wall.2 bottles of beer on the wall,2 bottles of beer.Take one down, pass it around,one bottle of beer on the wall.one bottle of beer on the wall,one bottle of beer.Take one down, pass it around,no more bottles of beer on the wall." | 5 bottles of beer on the wall,5 bottles of beer.Take one down, pass it around,4 bottles of beer on the wall.4 bottles of beer on the wall,4 bottles of beer.Take one down, pass it around,3 bottles of beer on the wall.3 bottles of beer on the wall,3 bottles of beer.Take one down, pass it around,2 bottles of beer on the wall.2 bottles of beer on the wall,2 bottles of beer.Take one down, pass it around,one bottle of beer on the wall.one bottle of beer on the wall,one bottle of beer.Take one down, pass it around,no more bottles of beer on the wall. |