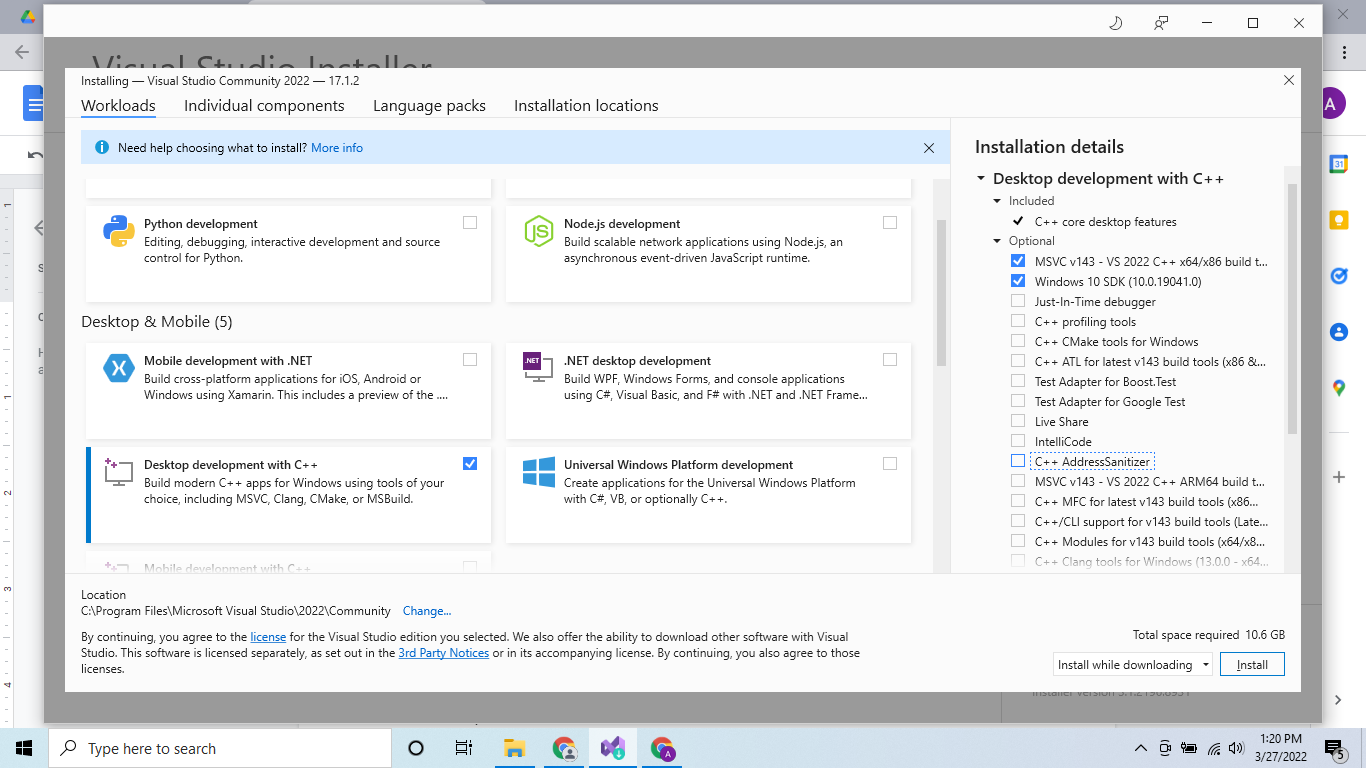
1. This document is for two way python and c++ module demo. Setup is step1.
   1. The setup is done first by installing visual studio 2019 and setting system environment variables.
      1. Install visual studio 2019 community <<https://www.youtube.com/watch?v=pjYq6p42QTc>>
      2. Chose as below.



* + 1. The machine should have Visual studio community 2019 and the variable paths be defiled in SYSTEM variables

* + - 1. C:\Program Files (x86)\Windows Kits\10\bin\10.0.19041.0\x64 (binary)
      2. C:\Program Files (x86)\Microsoft Visual Studio\2019\Community\VC\Tools\MSVC\14.29.30133\bin\Hostx64\x64
      3. E:\AnacondaFolder (where python is installed)
      4. E:\AnacondaFolder\Scripts

1. After Setup, this section is about steps needed to create c and pyd modules
   1. In python pip install cython in your python virtual env.
   2. Activate the virtual environment and install jupyter <pip install jupyter> ,as we have to call the .pyd file from jupyter also
   3. open RE file in python which has functions written in python and save it with .pyx extension.

---------- RE.py

def fn():

print("Hi")

fn()

----------

And

---------- RE.pyx

def fn():

print("Hi")

fn()

----------

* 1. Create a setup.py file

---------- setup.py

from distutils.core import setup

from Cython.Build import cythonize

directives = {'linetrace':False, 'language\_level':3}

setup(ext\_modules = cythonize ('RE.pyx'))

---------- setup.py

* 1. activate virtual environment.
  2. Cd to this file location and open a cmd and activate your environment and run python setup.py build\_extn –inplace
  3. There will be few files and build folder created. Only .pyd file is needed and it has a long name. Delete the long name and keep it as RE.pyd in any folder
  4. In pycharm create a calling .py file and import RE after placing RE.pyd file in same location as calling program and it will work
  5. Now from terminal of Pycharm, launch jupyter and and create a new notebook and run cell <import RE>

1. This section details how to use a CPP module in python
   1. In this section we have to build our understanding of files like (pyd, pyx, pxd,cpp)
      1. Setup.py ←– jarvis\_report.pyx
      2. Jarvis\_report.pyx ←- jarvisreport.cpp ←- jarvisreport.h
      3. Jarvis\_report.pyx ←- jarvisreport.pxd ←- jarvisreport.h
   2. The setup.py file has
      1. cythonize([“jarvis\_report.pyx”])
   3. The .pyx file
      1. # distutils: language=c++
      2. # distutils:source = jarvisreport.cpp
      3. As cython and python function definitions
      4. Calls .pxd file where cython cppdef style function signature
      5. Name of .cpp and .pxd files should be exactly same with signature in pyx, and .pxd files should be exactly same.
   4. The .pxd file
      1. Has cython syntax function signatures
      2. Only signature.. No implementation.
      3. Cython syntax (cppclass)
      4. Has cdef extern from jarvisreport.h
   5. The .cpp file
      1. It is a must
      2. Implementation defined here
      3. Calls .h file
   6. The .h file
      1. All classes and structs are defined here