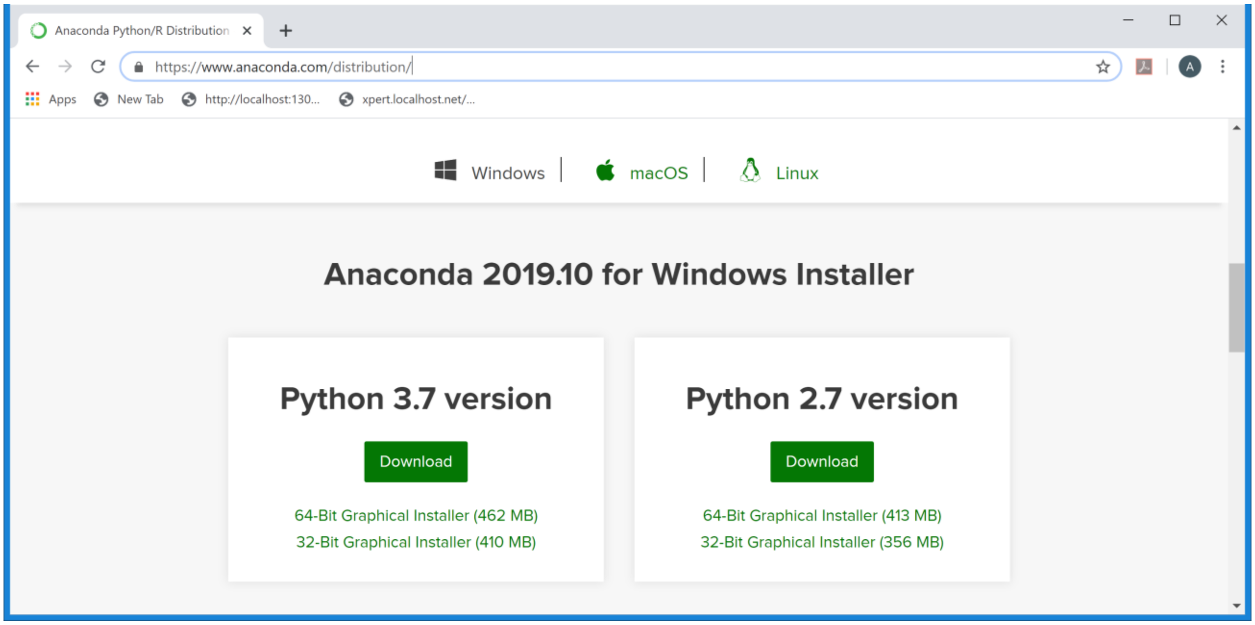
Programming Logistic Regression Using Python

This handout shows the Logistic Regression Python code under Visual Studio. If you wanted to use some other Python development environment, then the code is still valid. If you are new to Python, then first make sure that yu have Visual Studio 2019 Community edition installed (it is a free download), and also make sure when you install Visual Studio, you choose the Python development option along with other options.

Setting up the Python Environment:

Download the latest Python version (3.7 as of this writing) from the following url: https://www.anaconda.com/distribution/


Anaconda Python/R Distribution
https://www.anaconda.com/distribution/
Apps
New Tab
http://localhost:130...
xpert.localhost.net/...
Windows
macOS
Linux
Anaconda 2019.10 for Windows Installer
Python 3.7 version
Download
64-Bit Graphical Installer (462 MB)
32-Bit Graphical Installer (410 MB)
Python 2.7 version
Download
64-Bit Graphical Installer (413 MB)
32-Bit Graphical Installer (356 MB)

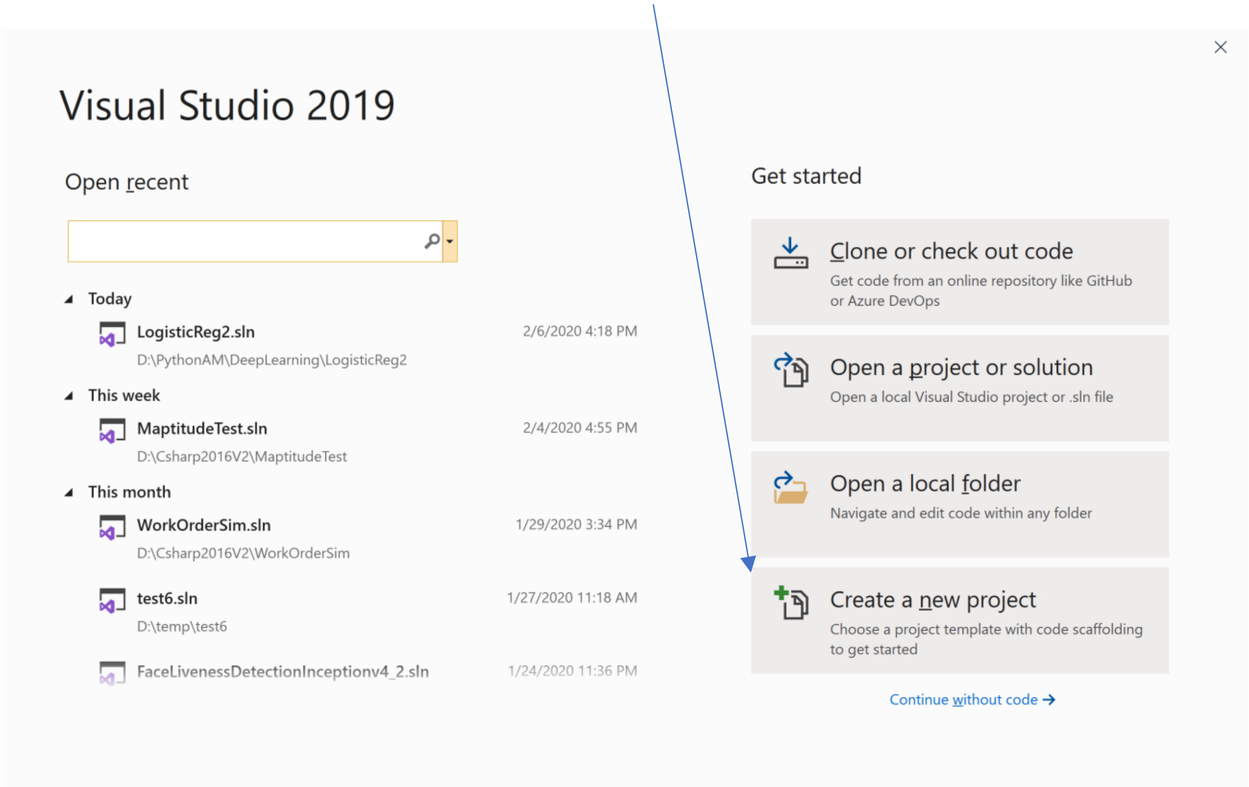
As you install Python, it will give you an option to set the path to the python environment, make sure you check this checkbox even if it gives you a warning.

Then launch command prompt as an administrator and type the following commands (one at a time):

```
python -m pip install --upgrade pip
conda create -n tensorflow2cpu pip python=3.7
activate tensorflow2cpu
pip install --upgrade tensorflow
```

If the above commands run successfully, your Python environment is ready with the latest Tensorflow library.

Launch Visual studio and create a new project as shown below.



Visual Studio 2019

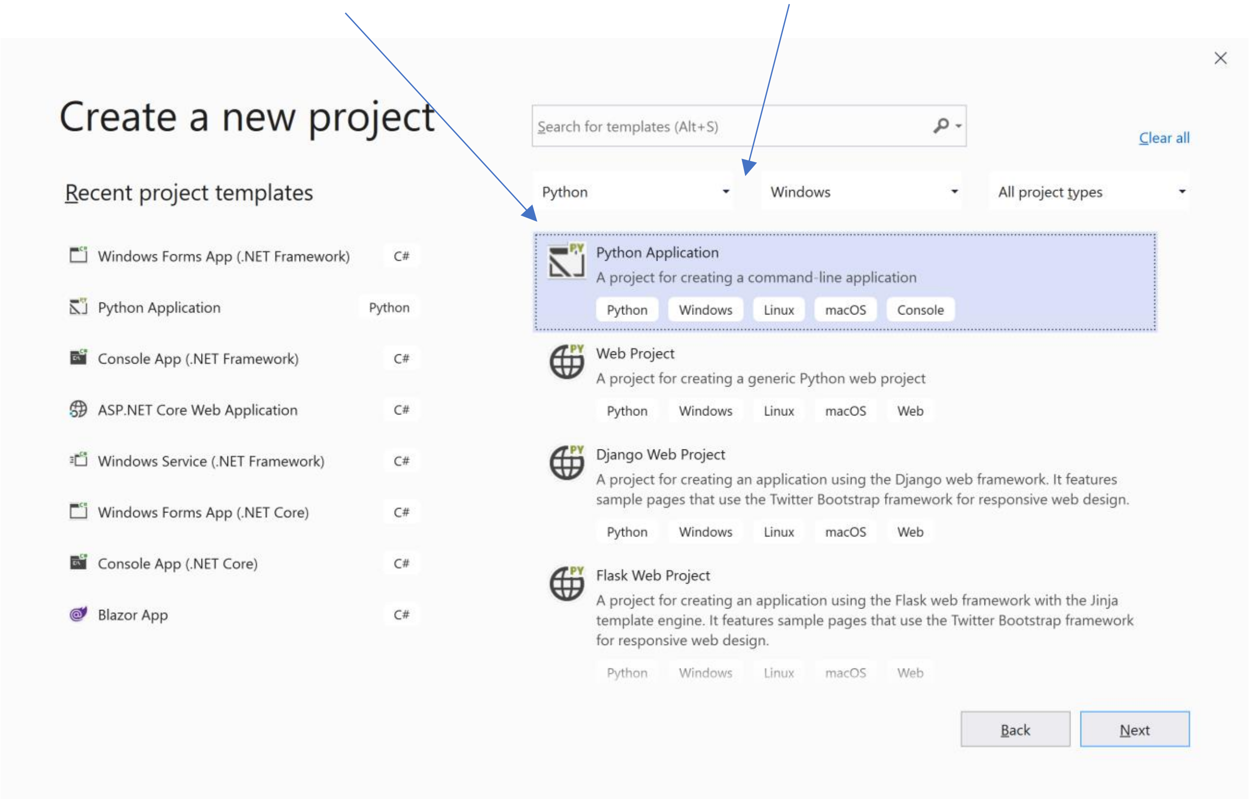
Open recent

- Today
  - LogisticReg2.sln — D:\PythonAM\DeepLearning\LogisticReg2 — 2/6/2020 4:18 PM
- This week
  - MaptitudeTest.sln — D:\Csharp2016V2\MaptitudeTest — 2/4/2020 4:55 PM
- This month
  - WorkOrderSim.sln — D:\Csharp2016V2\WorkOrderSim — 1/29/2020 3:34 PM
  - test6.sln — D:\temp\test6 — 1/27/2020 11:18 AM
  - FaceLivenessDetectionInceptionv4_2.sln — 1/24/2020 11:36 PM

Get started

- Clone or check out code — Get code from an online repository like GitHub or Azure DevOps
- Open a project or solution — Open a local Visual Studio project or .sln file
- Open a local folder — Navigate and edit code within any folder
- Create a new project — Choose a project template with code scaffolding to get started

Choose the following on the next screen (select Python from the dropdown), then click Next:



Create a new project

Python | Windows | All project types

Recent project templates

Windows Forms App (.NET Framework)	C#
Python Application	Python
Console App (.NET Framework)	C#
ASP.NET Core Web Application	C#
Windows Service (.NET Framework)	C#
Windows Forms App (.NET Core)	C#
Console App (.NET Core)	C#
Blazor App	C#

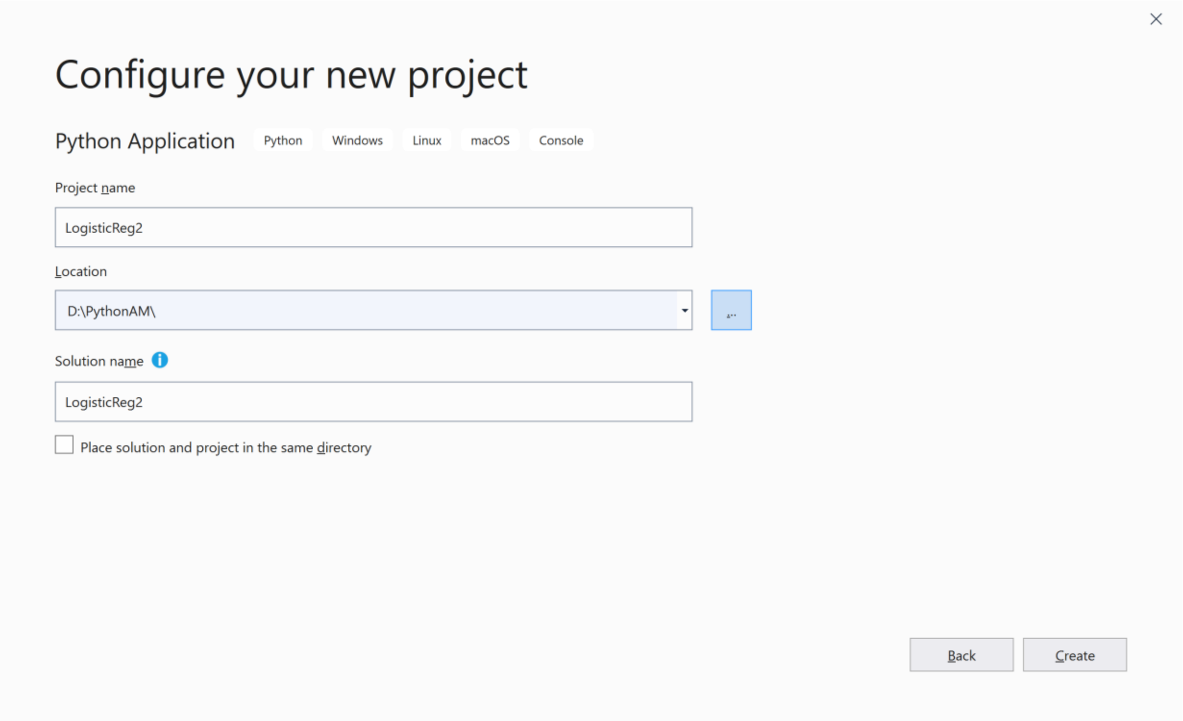
Python Application
A project for creating a command-line application
Python Windows Linux macOS Console

Web Project
A project for creating a generic Python web project
Python Windows Linux macOS Web

Django Web Project
A project for creating an application using the Django web framework. It features sample pages that use the Twitter Bootstrap framework for responsive web design.
Python Windows Linux macOS Web

Flask Web Project
A project for creating an application using the Flask web framework with the Jinja template engine. It features sample pages that use the Twitter Bootstrap framework for responsive web design.
Python Windows Linux macOS Web

Give a project name of LogisticReg2, and select the directory where you would like the project to be created. Then click on the Create button as shown below



Configure your new project

Python Application · Python · Windows · Linux · macOS · Console

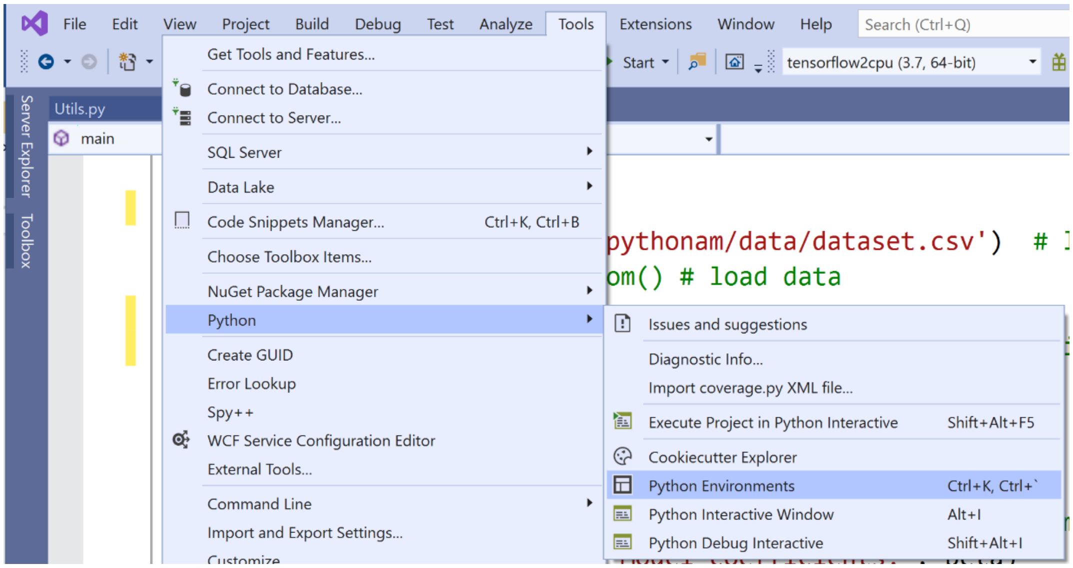
Project name: LogisticReg2

Location: D:\PythonAM\

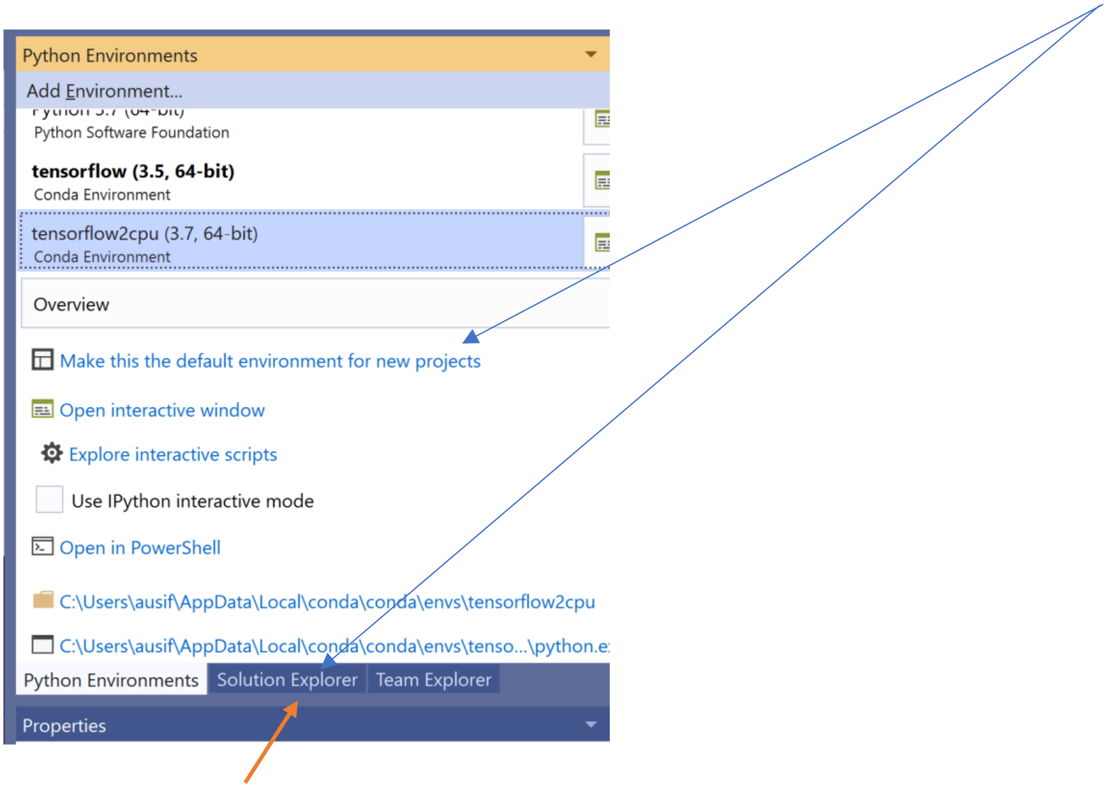
Solution name: LogisticReg2

□ Place solution and project in the same directory

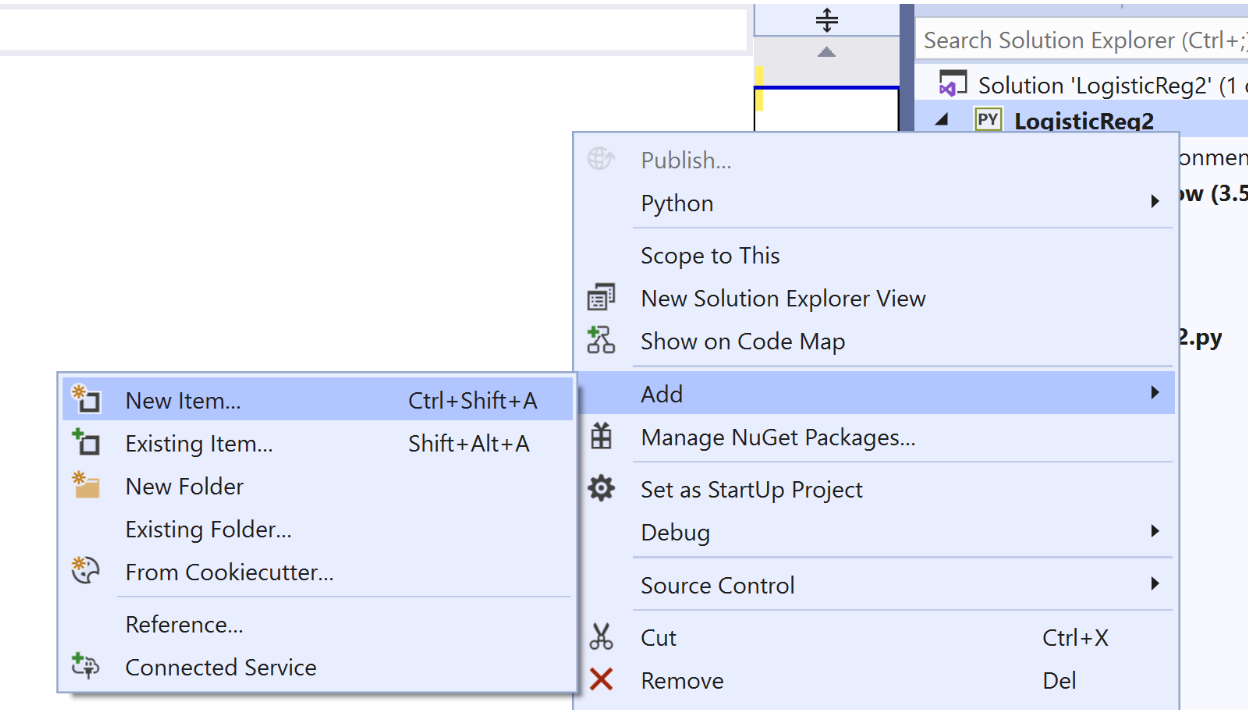
From the Tools menu, select Python-> Python Environments.



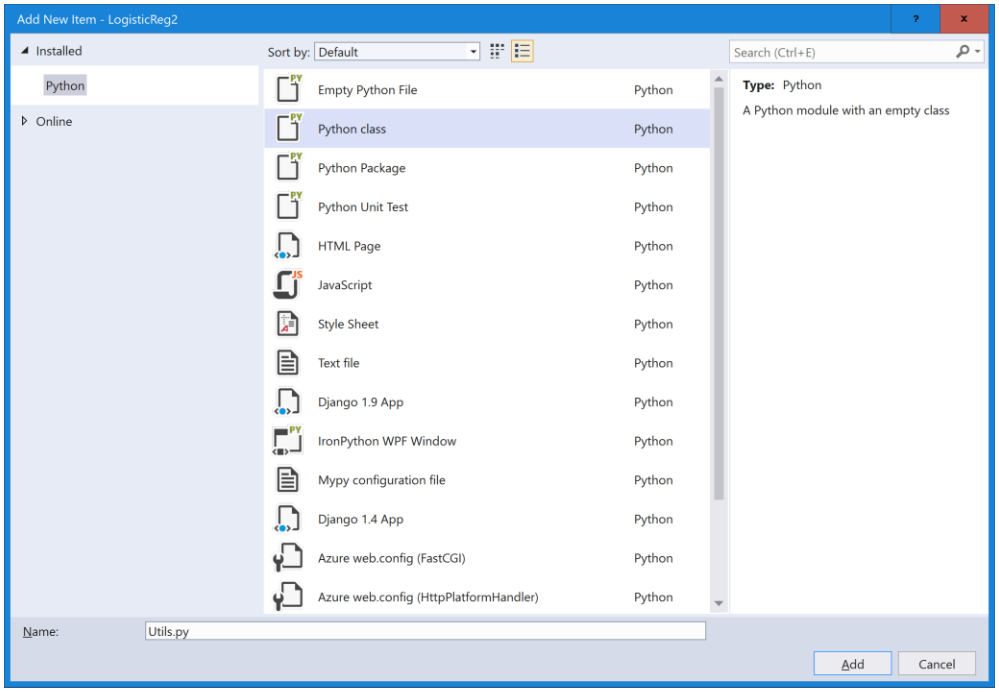
Then make the Tensorflow2cpu as the default environment for running Python, as shown below.


Python Environments
Add Environment...
Python 3.7 (64-bit)
Python Software Foundation
tensorflow (3.5, 64-bit)
Conda Environment
tensorflow2cpu (3.7, 64-bit)
Conda Environment
Overview
Make this the default environment for new projects
Open interactive window
Explore interactive scripts
Use IPython interactive mode
Open in PowerShell
C:\Users\ausif\AppData\Local\conda\conda\envs\tensorflow2cpu
C:\Users\ausif\AppData\Local\conda\conda\envs\tenso...\python.e
Python Environments
Solution Explorer
Team Explorer
Properties

Then click on Solution Explorer tab, and right click on the project name, and choose Add Python class.



Then choose Python class as shown below. Name the class Utils.py.



Empty Python File	Python
Python class	Python
Python Package	Python
Python Unit Test	Python
HTML Page	Python
JavaScript	Python
Style Sheet	Python
Text file	Python
Django 1.9 App	Python
IronPython WPF Window	Python
Mypy configuration file	Python
Django 1.4 App	Python
Azure web.config (FastCGI)	Python
Azure web.config (HttpPlatformHandler)	Python

Type: Python

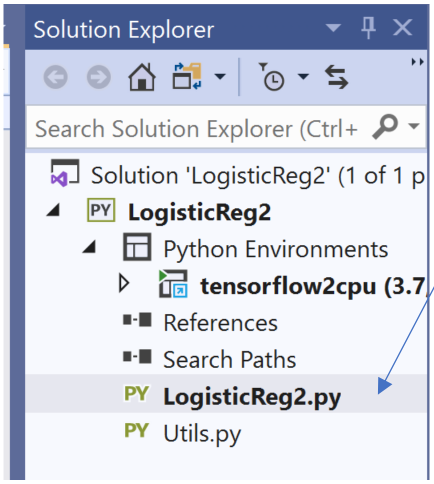
A Python module with an empty class

Name: Utils.py

Type the following code in Utils.py.

```
import csv
import numpy as np
import matplotlib.pyplot as plt

class Utils(object):
    def readData(self,filename): # return as numpy array
        with open(filename,"r") as csvfile:
            lines = csv.reader(csvfile)
            data = list(lines)
            for i in range(len(data)): # convert data to float
                data[i] = [float(x) for x in data[i]]
        return np.array(data)

    def readDataRandom(self):
        np.random.seed(12)
        num_observations = 50
        x1 = np.random.multivariate_normal([1.5, 4], [[1, .75],[.75, 1]], 
num_observations)
        print(x1.shape)
        x2 = np.random.multivariate_normal([1, 2], [[1, .75],[.75, 1]], 
num_observations)
        data = np.vstack((x1, x2)).astype(np.float32)
        print(data.shape)
        labels = np.hstack((np.zeros(num_observations), np.ones(num_observations)))
        print(labels.shape)
        dataWithLabels = np.hstack((data,labels.reshape(labels.shape[0],1)))
        #print(dataWithLabels.shape)
        #print(labels.shape)
        plt.figure(figsize=(12,8))
        plt.scatter(data[:, 0], data[:, 1], c = labels, alpha = .4)
        plt.show()
        return dataWithLabels

    def normalizeData(self,X):
        min = np.min(X, axis = 0)
        max = np.max(X, axis = 0)
        normX = 1 - ((max - X)/(max-min))
        return normX

    def plot_result(self,X, y, beta):
        x_0 = X[np.where(y == 0.0)]
        x_1 = X[np.where(y == 1.0)]

        # plot the data points
        plt.scatter([x_0[:, 1]], [x_0[:, 2]], c='b', label='y = 0')
        plt.scatter([x_1[:, 1]], [x_1[:, 2]], c='r', label='y = 1')

        # plot the decision boundary
        x1 = np.arange(0, 1, 0.1)
        x2 = -(beta[0,0] + beta[0,1]*x1)/beta[0,2]
        plt.plot(x1, x2, c='g', label='reg line')

        plt.xlabel('x1')
        plt.ylabel('x2')
        plt.legend()
        plt.show()
```

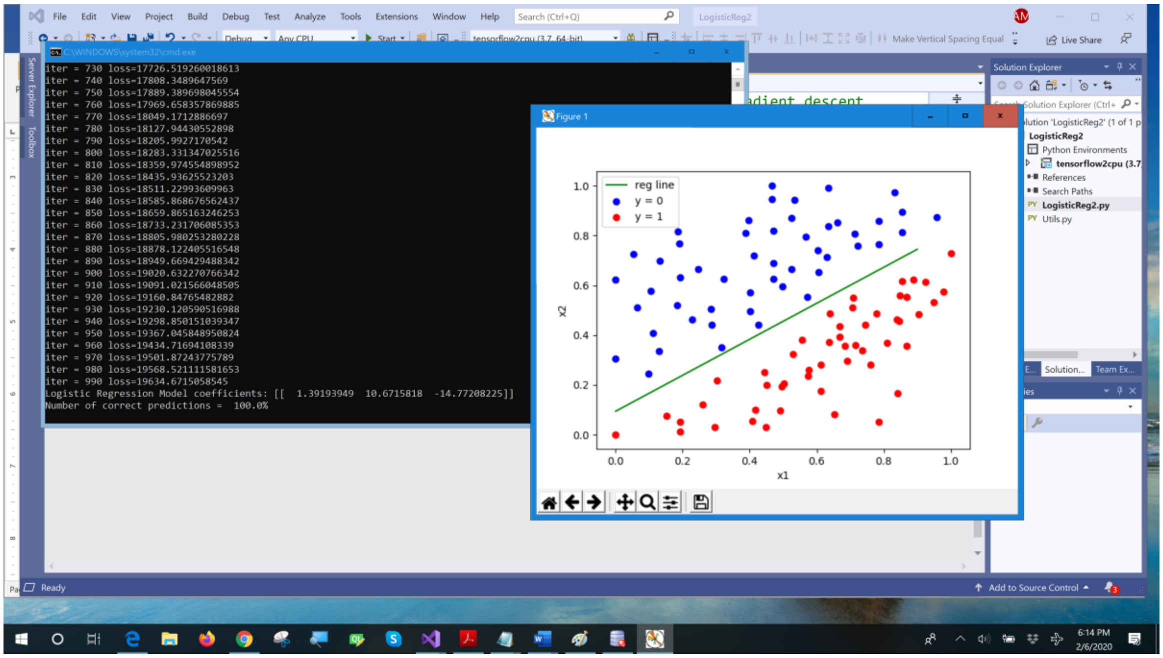
From the solution explorer, double click on the LogisticReg2.py and type the following code in it.



```
import sys
from Utils import Utils
import numpy as np

def sigmoid(beta, X):  # logistic function
    return 1.0/(1 + np.exp(-np.dot(X, beta.T)))

def gradientBeta(beta, X, y):   # dL/dbeta
    a = sigmoid(beta, X)
    part1 = a - y.reshape(X.shape[0], 1)
    grad = np.dot(part1.T, X)
    return grad
```

```
def logLoss(beta, X, y):
        a = sigmoid(beta, X) # actual output
        loss = -(y * np.log(a)  + (1 - y) * np.log(1 - a))
        return np.sum(loss)

def trainUsingGradientDescent(X, y, beta, num_iter, alpha = .01):
        loss = logLoss(beta, X, y)
        for i in range (num_iter):
                beta = beta - (alpha * gradientBeta(beta, X, y))
                loss = logLoss(beta, X, y)
                if (i%10 == 0):
                        print('iter = ' + str(i) + ' loss=' + str(loss))
        return beta


def classifyData(beta, X):  # 0 or 1
        a = sigmoid(beta, X) # actual output
        decision = np.where(a >= .5, 1, 0)
        return decision


def main():
        utils = Utils()

        data = utils.readData('d:/pythonam/data/dataset.csv')  # load data
        #data = utils.readDataRandom() # load data

        X = utils.normalizeData(data[:,0:2])  # or [:,:-1] normalize data - scale between
0-1
        X = np.hstack((np.ones((1,X.shape[0])).T, X))    # add 1's column to data

        Y = data[:, -1] # expected output, -1 means last column
        beta = np.zeros((1,X.shape[1]))  # (1,3) in this example
        beta = trainUsingGradientDescent(X, Y, beta,1000)  # optimize using gradient
descent
        print("Logistic Regression Model coefficients:", beta)

        y_predicted = classifyData(beta, X)  # predictions by the trained model
        #print(y_predicted.shape)
        print("Number of correct predictions = ", str(np.sum(Y ==
y_predicted.reshape(Y.shape[0]))/len(X)*100) + '%')

        utils.plot_result(X, Y, beta) # plot results

if __name__ == "__main__":
    sys.exit(int(main() or 0))
```

From the Debug menu, choose Start without Debugging. The output will appear as:



```
iter = 730 loss=17726.519260018613
iter = 740 loss=17808.3489647569
iter = 750 loss=17889.389698045554
iter = 760 loss=17969.658357869885
iter = 770 loss=18049.1712886697
iter = 780 loss=18127.94430552898
iter = 790 loss=18205.9927170542
iter = 800 loss=18283.331347025516
iter = 810 loss=18359.974554898952
iter = 820 loss=18435.93625523203
iter = 830 loss=18511.22993609963
iter = 840 loss=18585.868676562437
iter = 850 loss=18659.865163246253
iter = 860 loss=18733.231706085353
iter = 870 loss=18805.980253280228
iter = 880 loss=18878.122405516548
iter = 890 loss=18949.669429488342
iter = 900 loss=19020.632270766342
iter = 910 loss=19091.021566048505
iter = 920 loss=19160.84765482882
iter = 930 loss=19230.120590516988
iter = 940 loss=19298.850151039347
iter = 950 loss=19367.045848950824
iter = 960 loss=19434.71694108339
iter = 970 loss=19501.87243775789
iter = 980 loss=19568.521111581653
iter = 990 loss=19634.6715058545
Logistic Regression Model coefficients: [[  1.39193949  10.6715818  -14.77208225]]
Number of correct predictions =  100.0%
```

```
adient descent
```