💥

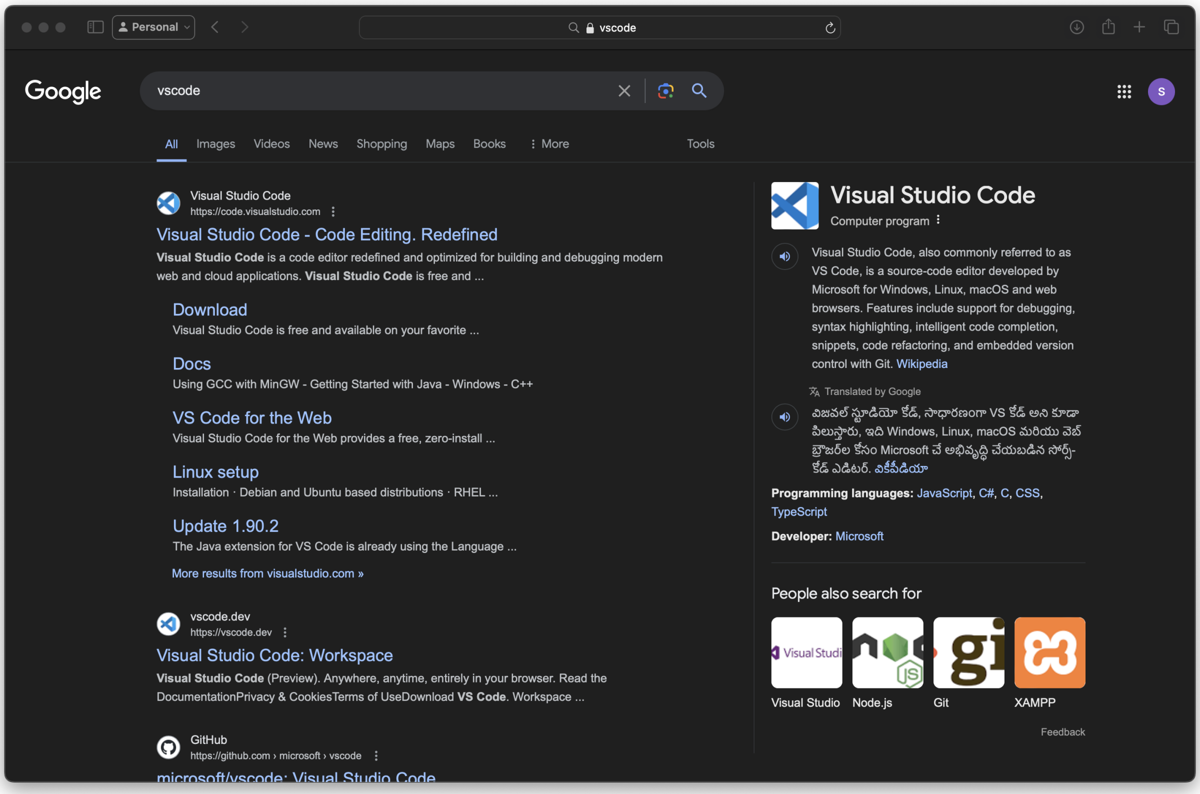
# PROJECT - M

---

IKA MODALEDUDAMA...?

## Mundu urgent ga google open chei , and akkada “ vs code download” ani otthu mama.

ippudu ah vs code ni download chesko.okkasari adi download ipoyaka next steps chepta!


Personal

vscode

Google

vscode

All Images Videos News Shopping Maps Books ⋮ More Tools

Visual Studio Code
https://code.visualstudio.com

Visual Studio Code - Code Editing. Redefined

**Visual Studio Code** is a code editor redefined and optimized for building and debugging modern web and cloud applications. **Visual Studio Code** is free and ...

Download
Visual Studio Code is free and available on your favorite ...

Docs
Using GCC with MinGW - Getting Started with Java - Windows - C++

VS Code for the Web
Visual Studio Code for the Web provides a free, zero-install ...

Linux setup
Installation · Debian and Ubuntu based distributions · RHEL ...

Update 1.90.2
The Java extension for VS Code is already using the Language ...

More results from visualstudio.com »

vscode.dev
https://vscode.dev

Visual Studio Code: Workspace

**Visual Studio Code** (Preview). Anywhere, anytime, entirely in your browser. Read the DocumentationPrivacy & CookiesTerms of UseDownload **VS Code**. Workspace ...

GitHub
https://github.com › microsoft › vscode

microsoft/vscode: Visual Studio Code

Visual Studio Code
Computer program

Visual Studio Code, also commonly referred to as VS Code, is a source-code editor developed by Microsoft for Windows, Linux, macOS and web browsers. Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded version control with Git. Wikipedia

Translated by Google

విజువల్ స్టూడియో కోడ్, సాధారణంగా VS కోడ్ అని కూడా పిలుస్తారు, ఇది Windows, Linux, macOS మరియు వెబ్ బ్రౌజర్‌ల కోసం Microsoft చే అభివృద్ధి చేయబడిన సోర్స్-కోడ్ ఎడిటర్. వికీపీడియా

**Programming languages:** JavaScript, C#, C, CSS, TypeScript

**Developer:** Microsoft

People also search for

Visual Studio Node.js Git XAMPP

Feedback

okasari download chesaka open cheyyandi.

oka welcome page ostadi adi close chei! ne desktop meda normal ga oka folder create chesi dani peru project ani pettu, and ah empty folder ni drag chesi vs code lo padesey!

ah tarwata ade chrome lo python download chesko!

like this ━

## STEP1:- Installing Python:

1. Open Chrome or any web browser.
2. In the search bar, type "Python download" and press Enter.
3. Click on the link that says "Download Python" (this should lead to the official Python website).

4. Choose the latest version of Python and download the installer.
5. Once the download is complete, open the downloaded file and follow the installation instructions:
    - On Windows: Run the installer, check "Add Python to PATH," and follow the setup wizard.
    - On Mac: Open the downloaded `.pkg` file and follow the installation instructions.
    - On Linux: Follow the provided instructions for your specific distribution.

## Step 2: Creating a New Project

**2.1 Creating a New Folder:**

1. Open the File Explorer (Windows), Finder (Mac), or your file manager (Linux).
2. Navigate to a location where you want to create your project folder (e.g., Documents).
3. Right-click (or Control-click on Mac) and select "New Folder."
4. Name the folder `project`.

**2.2 Opening the Folder in VS Code:**

1. Open Visual Studio Code.
2. Click on "File" in the top menu and select "Open Folder."
3. Navigate to the `project` folder you created and open it.

## Step 3: Installing Necessary Libraries

**3.1 Opening the Terminal in VS Code:**

1. In Visual Studio Code, click on "Terminal" in the top menu.
2. Select "New Terminal" from the dropdown.

### 3.2 Installing Libraries:

1. In the terminal, type the following commands and press Enter after each one:

```
pip install numpy pandas matplotlib tensorflow pillow
```

## Step 4: Loading the MNIST Dataset

### 4.1 Creating a New Python File:

1. In Visual Studio Code, click on the "New File" icon or press `Ctrl+N` (Cmd+N on Mac).
2. Name the file `digits.py` and save it.

### 4.2 Writing Code to Load the Dataset:

1. Open the `digits.py` file.

- Copy and paste the following code:

```
import ssl
ssl._create_default_https_context = ssl._create_unverif
ied_context

import tensorflow as tf
from tensorflow.keras.datasets import mnist
import matplotlib.pyplot as plt
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten
from PIL import Image, ImageOps
import numpy as np
```

deni ardam entidi ante:-

## Code Explanation

```
import ssl
ssl._create_default_https_context = ssl._create_unverified
_context

import tensorflow as tf
from tensorflow.keras.datasets import mnist
import matplotlib.pyplot as plt
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten
from PIL import Image, ImageOps
import numpy as np
```

**Explanation:**

1. **SSL Context:**
    - The first two lines configure SSL to ignore certificate verification errors. This ensures that the script can download datasets from the internet without encountering SSL errors.
2. **Import Libraries:**
    - `tensorflow`: The main library used for building and training the machine learning model.
    - `mnist`: The MNIST dataset, which contains images of handwritten digits.

- `matplotlib.pyplot`: A plotting library used to display images.
- `Sequential`, `Dense`, `Flatten`: These are components from TensorFlow Keras used to build and structure the neural network.
- `PIL`, `Image`, `ImageOps`: Libraries used for image processing.
- `numpy`: A library for numerical operations, particularly for handling arrays.

## ah tarwata

```
# Load the dataset
(x_train, y_train), (x_test, y_test) = mnist.load_data()
```

**Explanation:**

1. **Load the Dataset:**
    - This line downloads and loads the MNIST dataset.
    - `x_train`, `y_train`: Training images and their corresponding labels.
    - `x_test`, `y_test`: Testing images and their corresponding labels.

next edi raayi mawa

```
# Normalize the data
x_train, x_test = x_train / 255.0, x_test / 255.0
```

**Explanation:**

1. **Normalize the Data:**

- Each pixel value in the images ranges from 0 to 255. Dividing by 255.0 normalizes these values to a range of 0 to 1, making the data easier for the model to process.

---

next edi :-

```
# Display the first image in the training dataset
plt.imshow(x_train[0], cmap='gray')
plt.show()
```

**Explanation:**

1. **plt.imshow(x_train[0], cmap='gray'):**
    - This line tells our drawing tool (matplotlib) to show the first picture in our training set.
    - `x_train[0]` is the first picture.
    - `cmap='gray'` means we want to show the picture in shades of gray (like a black-and-white photo).
2. **plt.show():**
    - This command tells the drawing tool to display the picture on the screen.
    - When we run this part of the code, we should see a picture of a handwritten number.

## Recap

- We brought in special tools and a box of handwritten numbers.

- We opened the box and separated the pictures and answers into training and testing sets.
- We adjusted the brightness of the pictures to make it easier for the computer to learn.
- Finally, we displayed the first picture to see what it looks like.

By following these steps, we're getting everything ready to teach the computer how to recognize handwritten numbers!

---

## Step 5: Building the Model

### 5.1 Writing Code to Build a Neural Network:

1. Add the following code to `digits.py`:

```
# Build the model
model = Sequential([
    Flatten(input_shape=(28, 28)),
    Dense(128, activation='relu'),
    Dense(10, activation='softmax')
])

# Compile the model
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentrop
y',
              metrics=['accuracy'])
```

## deni clear ardam chepta vinu :

## Code Explanation

**Explanation:**

1. **from tensorflow.keras.models import Sequential:**
    - Imagine `Sequential` as a recipe book where we list the steps to create our machine learning model. It's like telling the computer the order in which to do things.
2. **from tensorflow.keras.layers import Dense, Flatten:**
    - `Dense` and `Flatten` are ingredients in our recipe.
    - `Dense` is like a layer of brain cells (neurons) that helps the computer learn.
    - `Flatten` is a tool that helps turn our 2D picture into a 1D list of numbers so the computer can process it easily.

```
# Build the model
model = Sequential([
    Flatten(input_shape=(28, 28)),
    Dense(128, activation='relu'),
    Dense(10, activation='softmax')
])
```

**Explanation:**

1. **model = Sequential([ ... ]):**
    - We are starting our recipe to build the model. We tell the computer to follow these steps in sequence.
2. **Flatten(input_shape=(28, 28)):**
    - The first step is to flatten the 28×28 picture (like unfolding a piece of paper into a long list). This helps the computer read the picture more easily.

3. **Dense(128, activation='relu'):**
    - Next, we add a layer of 128 neurons. This layer helps the computer learn patterns from the picture.
    - `activation='relu'` is like telling the neurons to wake up and work in a specific way. `ReLU` (Rectified Linear Unit) helps the neurons understand patterns better.
4. **Dense(10, activation='softmax'):**
    - Finally, we add a layer with 10 neurons. Each neuron represents a number from 0 to 9.
    - `activation='softmax'` helps the computer decide which number the picture most likely represents. It's like voting and picking the number with the most votes.

---

```
# Compile the model
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentro
py',
              metrics=['accuracy'])
```

**Explanation:**

1. **model.compile( ... ):**
    - Now, we need to tell the computer how to learn and measure its progress.
2. **optimizer='adam':**
    - `Adam` is like a smart coach that helps the computer learn faster and better. It adjusts the learning process to improve the model's performance.
3. **loss='sparse_categorical_crossentropy':**

- `loss` is like the computer's way of measuring how wrong it is. `Sparse categorical crossentropy` is a method to calculate this error. It helps the computer understand how to improve its guesses.

4. **metrics=['accuracy']:**
   - We also want to know how many times the computer gets the right answer. `Accuracy` is the measure of how often the computer's guess matches the correct number.

---

## Recap

- We brought in tools and ingredients to build our model.
- We created a sequence of steps (layers) for the computer to follow.
- We flattened the 28×28 picture into a list of numbers.
- We added two layers: one with 128 neurons to learn patterns, and one with 10 neurons to decide which number the picture represents.
- We set up a coach (`Adam`) and a method to measure mistakes (`loss`) and successes (`accuracy`).

By following these steps, we've built a simple neural network model that can learn to recognize handwritten numbers!

4o

## Step 6: Training the Model

## ippudu ne code lo edi add chei:

### 6.1 Writing Code to Train the Model:

1. Add the following code to `digits.py`:

```
# Train the model
model.fit(x_train, y_train, epochs=10)
```

2. **Explanation:**
   - `fit` trains the model using the training data for 5 epochs (iterations over the dataset).

## ah tyarwata edi add chei:

## Step 7: Evaluating the Model

### 7.1 Writing Code to Evaluate the Model:

1. Add the following code to `digits.py`:

```
# Evaluate the model
loss, accuracy = model.evaluate(x_test, y_test)
print(f"Accuracy: {accuracy * 100:.2f}%")
```

2. **Explanation:**
   - `evaluate` tests the model using the testing data.
   - The accuracy of the model is printed.

## ah tarwata edi :

```
# Function to preprocess the image
def preprocess_image(image_path):
    img = Image.open(image_path).convert('L')
    img = ImageOps.invert(img)
    img = img.resize((28, 28))
    img = np.array(img) / 255.0
    img = img.reshape(1, 28, 28)
    return img
```

**Explanation:**

1. **Preprocess the Image:**
    - **Image.open(image_path).convert('L')**: Opens the image and converts it to grayscale.
    - **ImageOps.invert(img)**: Inverts the image colors (black to white and vice versa).
    - **img.resize((28, 28))**: Resizes the image to 28×28 pixels.
    - **np.array(img) / 255.0**: Converts the image to a NumPy array and normalizes pixel values to the range [0, 1].
    - **img.reshape(1, 28, 28)**: Reshapes the array to the format expected by the model.

## ah tarwata edi:-

```
# Path to the handwritten digit image
image_path = 'digit.png'
new_image = preprocess_image(image_path)
```

**Explanation:**

1. **Prepare the Test Image:**
    - The image path is set to `'digit.png'`.
    - The image is preprocessed using the `preprocess_image` function.

## ah tarwata edi :

```
# Predict the digit
prediction = model.predict(new_image)
predicted_digit = np.argmax(prediction)
print(f"Predicted Digit: {predicted_digit}")
```

**Explanation:**

1. **Predict the Digit:**
    - The preprocessed image is passed to the model for prediction.
    - `np.argmax(prediction)` finds the digit with the highest probability.
    - The predicted digit is printed.

---

```
# Display the test image and the predicted digit
```

```
plt.imshow(new_image.reshape(28, 28), cmap='gray')
plt.title(f"Predicted Digit: {predicted_digit}")
plt.show()
```

**Explanation:**

1. **Display the Prediction:**
    - The test image is displayed with the predicted digit as the title.
    - `new_image.reshape(28, 28)` reshapes the image back to 28×28 pixels for display.

## mottham ela untadi:

### Step 8: Testing the Model with New Data

#### 8.1 Writing Code to Preprocess and Predict Handwritten Digits:

anni step by step okko line of code raskuntu pothe ne code konchem ela kanabadtadi!:-

code in `digits.py`:

```



def preprocess_image(image_path):
    img = Image.open(image_path).convert('L')
    img = ImageOps.invert(img)
    img = img.resize((28, 28))
    img = np.array(img) / 255.0
    img = img.reshape(1, 28, 28)
    return img

# Path to the handwritten digit image
image_path = 'digit.png'
new_image = preprocess_image(image_path)

# Predict the digit
prediction = model.predict(new_image)
print(f"Predicted Digit: {np.argmax(prediction)}")
```

**Explanation:**

- `PIL` (Pillow) is used for image processing.
- `preprocess_image` function converts the image to grayscale, inverts it, resizes it to 28×28, normalizes it, and reshapes it for the model.
- The model predicts the digit from the processed image.

ippudu digits.py code lo :-

```


# Path to the handwritten digit image
image_path = 'digit.png'
new_image = preprocess_image(image_path)

```

e part daggara ‘digit.png’ unnadi kada so nuv em chestav ante oka pen paper teskoni

0 nundi 9 daka oka 5 numbers rai. and danni photo tesi digit.png ani perutho ne project folder lo padey!

## last ki nuv rasina code mottham ela kanabadtadi! :-

`digits.py`:

```
import ssl
ssl._create_default_https_context = ssl._create_unverified_co
ntext

import tensorflow as tf
from tensorflow.keras.datasets import mnist
import matplotlib.pyplot as plt
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten
from PIL import Image, ImageOps
import numpy as np

# Load the dataset
(x_train, y_train), (x_test, y_test) = mnist.load_data()

# Normalize the data
x_train, x_test = x_train / 255.0, x_test / 255.0

# Display the first image in the training dataset
plt.imshow(x_train[0], cmap='gray')
plt.show()

# Build the model
model = Sequential([
    Flatten(input_shape=(28, 28)),
    Dense(128, activation='relu'),
    Dense(10, activation='softmax')
])

# Compile the model
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

# Train the model
model.fit(x_train, y_train, epochs=10)  # Increase the number
of epochs

# Evaluate the model
loss, accuracy = model.evaluate(x_test, y_test)
print(f"Accuracy: {accuracy * 100:.2f}%")

# Function to preprocess the image
def preprocess_image(image_path):
    img = Image.open(image_path).convert('L')
    img = ImageOps.invert(img)
    img = img.resize((28, 28))
    img = np.array(img) / 255.0
    img = img.reshape(1, 28, 28)
    return img

# Path to the handwritten digit image
image_path = 'digit.png'
new_image = preprocess_image(image_path)

# Predict the digit
prediction = model.predict(new_image)
predicted_digit = np.argmax(prediction)
print(f"Predicted Digit: {predicted_digit}")

# Display the test image and the predicted digit
```

```
plt.imshow(new_image.reshape(28, 28), cmap='gray')
plt.title(f"Predicted Digit: {predicted_digit}")
plt.show()
```

## Explanation:

1. **Importing Libraries:**
    - We import all the necessary libraries for our project, including `tensorflow`, `mnist`, `matplotlib`, `Sequential`, `Dense`, `Flatten`, `PIL`, and `numpy`.
2. **Loading the Dataset:**
    - We load the MNIST dataset, which contains handwritten digit images and their labels.
3. **Normalizing the Data:**
    - We normalize the image pixel values to be between 0 and 1.
4. **Displaying the First Image:**
    - We display the first image from the training dataset to see what it looks like.
5. **Building the Model:**
    - We build a simple neural network model with a `Flatten` layer to convert images to 1D arrays, a `Dense` layer with 128 neurons, and another `Dense` layer with 10 neurons for classification.
6. **Compiling the Model:**
    - We compile the model with the Adam optimizer, sparse categorical cross-entropy loss function, and accuracy metric.
7. **Training the Model:**
    - We train the model using the training data for 5 epochs.
8. **Evaluating the Model:**

- We evaluate the model's performance using the test data and print the accuracy.

9. **Preprocessing the Image:**
   - We define a function to preprocess a new handwritten digit image by converting it to grayscale, inverting it, resizing it to 28×28 pixels, normalizing the pixel values, and reshaping it.
10. **Predicting the Digit:**
    - We use the model to predict the digit in the new image and print the predicted digit.

By running this complete code, you will be able to train a neural network on the MNIST dataset, evaluate its performance, and use it to predict new handwritten digit images.

## Ippudu the best part! nuv motthaniki oka machine learning model ni create chesav, with neural networks .... as simple as that!

### ippudu test cheddama?

terminal open chei :-

left right corner lo untadi, for mac users left lo top lo file , edit avanni untai kada akkada terminal untadi!

open chesaka oka environment ni create chei ela ante:-

terminal lo e code kottu:-

```
python3 -m venv env
```

ah tarwata activate cheyyali, ela ante :

## Activate the Virtual Environment

After creating the virtual environment, you need to activate it. The command to activate the virtual environment depends on your operating system.

- **On Windows:**

  ```
  .\env\Scripts\activate
  ```

- **On Mac/Linux:**

  ```
  source env/bin/activate
  ```

  e command kodite occhestadi!

  and now lets test it

## Run Your Python Script:

```
python digits.py
```

make sure edi run chese munde nuv ne project folder lo white paper meda oka number ni rasi digit.png ane name tho save chesi pettali! ah tarwate run cheyyali, cheyyagane oka image ostadi adi close cheyyagane modelk automatic ga train iyyi oka number ni predict chestadi!

the more you train , the more accurate it gets in predicting it! simple as that!

> NUV EVVARINA KANI, DOCTOR EY KANI LEDA EVVARINA KANI, YOU CAN LEARN ANY SKILL KASTHA OPIKA AND KASI UNTE CHALU!