**10 Python Projects with Code**

[[Evenword](https://medium.com/@Evenword?source=post_page-----cb13bdcbda39--------------------------------)](https://medium.com/@Evenword?source=post_page-----cb13bdcbda39--------------------------------)

[Evenword](https://medium.com/@Evenword?source=post_page-----cb13bdcbda39--------------------------------)

·

[Follow](https://medium.com/m/signin?actionUrl=https%3A%2F%2Fmedium.com%2F_%2Fsubscribe%2Fuser%2F7f53d24b78a4&operation=register&redirect=https%3A%2F%2Fmedium.com%2F%40Evenword%2F10-python-projects-with-code-cb13bdcbda39&user=Evenword&userId=7f53d24b78a4&source=post_page-7f53d24b78a4----cb13bdcbda39---------------------post_header-----------)

9 min read

·

Aug 3

406

9

10 Python Projects with Code to Enhance Your Programming Skills & You can mention in your Resume.

**Introduction**:  
Python is a versatile programming language known for its simplicity and readability. It is widely used for web development, data analysis, machine learning, and automation. One of the best ways to improve your Python skills is by working on practical projects. In this blog post, we will explore ten Python projects with code that will help you enhance your programming abilities. These projects cover a range of topics and difficulty levels, allowing you to grow as a Python developer. So, let’s dive into these exciting projects!

**1- URL Shortener:**

A URL shortener is a handy tool to condense long website links into shorter ones. In this project, you will build a URL shortener using Python and Flask, a popular web framework. By leveraging the power of Flask, you will learn how to handle HTTP requests, generate unique short codes, and redirect users to the original URL.

from flask import Flask, redirect, render\_template, request  
import string  
import random

app = Flask(\_\_name\_\_)# Dictionary to store the mappings of short codes to original URLs  
url\_mapping = {}  
def generate\_short\_code():  
 """Generate a random short code."""  
 characters = string.ascii\_letters + string.digits  
 short\_code = ''.join(random.choice(characters) for \_ in range(6))  
 return short\_code  
@app.route('/', methods=['GET', 'POST'])  
def home():  
 if request.method == 'POST':  
 original\_url = request.form['url']  
 short\_code = generate\_short\_code() url\_mapping[short\_code] = original\_url short\_url = request.host\_url + short\_code  
 return render\_template('index.html', short\_url=short\_url) return render\_template('index.html')  
@app.route('/<short\_code>')  
def redirect\_to\_original\_url(short\_code):  
 if short\_code in url\_mapping:  
 original\_url = url\_mapping[short\_code]  
 return redirect(original\_url)  
 else:  
 return "Short URL not found."  
if \_\_name\_\_ == '\_\_main\_\_':  
 app.run(debug=True)

*Also check👉:*[*200 E-books pack download in cheap cost limited time offer*](https://tinyurl.com/200bestEbooks)

**2. Image Caption Generator:**

Image captioning is a fascinating application of deep learning. In this project, you will use Python and the TensorFlow library to create an image caption generator. By combining computer vision and natural language processing techniques, your program will be able to generate descriptive captions for images automatically.

import tensorflow as tf  
import matplotlib.pyplot as plt  
import numpy as np  
from PIL import Image  
import os

# Load the pre-trained InceptionV3 model  
inception\_model = tf.keras.applications.InceptionV3(include\_top=True, weights='imagenet')# Load the tokenizer  
tokenizer = tf.keras.preprocessing.text.Tokenizer()  
tokenizer\_path = 'tokenizer.pkl'  
tokenizer = tf.keras.preprocessing.text.tokenizer\_from\_json(tokenizer\_path)# Define the maximum sequence length (number of words) for captions  
max\_sequence\_length = 20# Load the pre-trained caption generation model  
model\_path = 'caption\_generator\_model.h5'  
model = tf.keras.models.load\_model(model\_path)# Load the word-to-index and index-to-word mappings  
word\_to\_index = tokenizer.word\_index  
index\_to\_word = {index: word for word, index in word\_to\_index.items()}# Load the pre-trained InceptionV3 model  
inception\_model = tf.keras.applications.InceptionV3(include\_top=True, weights='imagenet')def preprocess\_image(image\_path):  
 """Preprocess the image for input to the InceptionV3 model."""  
 img = Image.open(image\_path)  
 img = img.resize((299, 299))  
 img = np.array(img)  
 img = img / 255.0  
 img = img.reshape(1, 299, 299, 3)  
 return imgdef generate\_caption(image\_path):  
 """Generate a caption for the given image."""  
 img = preprocess\_image(image\_path)  
 features = inception\_model.predict(img)  
 features = features.reshape(1, -1)  
   
 start\_token = tokenizer.word\_index['<start>']  
 end\_token = tokenizer.word\_index['<end>']  
   
 caption = []  
 input\_sequence = [start\_token]  
 for \_ in range(max\_sequence\_length):  
 sequence = np.array(input\_sequence)  
 y\_pred = model.predict([features, sequence])  
 y\_pred = np.argmax(y\_pred)  
   
 if index\_to\_word[y\_pred] == '<end>':  
 break  
   
 caption.append(index\_to\_word[y\_pred])  
 input\_sequence.append(y\_pred)  
   
 generated\_caption = ' '.join(caption)  
 return generated\_caption# Path to the image for caption generation  
image\_path = 'example\_image.jpg'# Generate caption for the image  
caption = generate\_caption(image\_path)  
print('Generated Caption:', caption)# Display the image  
img = Image.open(image\_path)  
plt.imshow(img)  
plt.axis('off')  
plt.show()

**3. Weather Forecast App:**

Building a weather forecast app will give you valuable experience in working with APIs. You will use Python and the OpenWeatherMap API to fetch weather data for a given location and display it to the user. This project will involve making HTTP requests, parsing JSON responses, and presenting the data in a user-friendly way.

import requests  
import json

def get\_weather\_data(api\_key, city):  
 """Get weather data for a specific city using the OpenWeatherMap API."""  
 base\_url = "http://api.openweathermap.org/data/2.5/weather"  
 params = {  
 "q": city,  
 "appid": api\_key,  
 "units": "metric"  
 }  
 response = requests.get(base\_url, params=params)  
 data = response.json()  
 return datadef display\_weather(data):  
 """Display weather information."""  
 if data["cod"] != "404":  
 city = data["name"]  
 country = data["sys"]["country"]  
 temperature = data["main"]["temp"]  
 description = data["weather"][0]["description"]  
 humidity = data["main"]["humidity"]  
 wind\_speed = data["wind"]["speed"] print(f"Weather in {city}, {country}:")  
 print(f"Temperature: {temperature}°C")  
 print(f"Description: {description}")  
 print(f"Humidity: {humidity}%")  
 print(f"Wind Speed: {wind\_speed} km/h")  
 else:  
 print("City not found. Please try again.")def main():  
 # API key from OpenWeatherMap  
 api\_key = "YOUR\_API\_KEY" # Get the city name from the user  
 city = input("Enter the city name: ") # Get weather data for the city  
 weather\_data = get\_weather\_data(api\_key, city) # Display weather information  
 display\_weather(weather\_data)if \_\_name\_\_ == "\_\_main\_\_":  
 main()

**4. Music Player:**

Creating a music player in Python is an excellent way to explore graphical user interfaces (GUIs). You can use the Tkinter library to design a basic music player that allows users to browse their music library, play songs, pause, stop, and adjust the volume. This project will help you gain insights into event-driven programming and GUI development.

import tkinter as tk  
import os  
from pygame import mixer

class MusicPlayer:  
 def \_\_init\_\_(self, root):  
 self.root = root  
 self.root.title("Music Player")  
 self.root.geometry("300x100") # Initialize Pygame mixer  
 mixer.init() # Create a variable to store the current playing status  
 self.playing = False # Create a variable to store the current selected song  
 self.current\_song = None # Create the UI elements  
 self.label = tk.Label(root, text="Music Player")  
 self.label.pack() self.play\_button = tk.Button(root, text="Play", command=self.play\_music)  
 self.play\_button.pack() self.stop\_button = tk.Button(root, text="Stop", command=self.stop\_music)  
 self.stop\_button.pack() self.browse\_button = tk.Button(root, text="Browse", command=self.browse\_music)  
 self.browse\_button.pack() def play\_music(self):  
 if self.current\_song:  
 if not self.playing:  
 mixer.music.load(self.current\_song)  
 mixer.music.play()  
 self.play\_button.config(text="Pause")  
 self.playing = True  
 else:  
 mixer.music.pause()  
 self.play\_button.config(text="Play")  
 self.playing = False def stop\_music(self):  
 mixer.music.stop()  
 self.play\_button.config(text="Play")  
 self.playing = False def browse\_music(self):  
 self.current\_song = tk.filedialog.askopenfilename(initialdir=os.getcwd(), title="Select Song",  
 filetypes=(("Audio Files", "\*.mp3"), ("All Files", "\*.\*")))  
 self.label.config(text=os.path.basename(self.current\_song))if \_\_name\_\_ == '\_\_main\_\_':  
 root = tk.Tk()  
 music\_player = MusicPlayer(root)  
 root.mainloop()

**5. Sudoku Solver:**

Solving Sudoku puzzles is a classic programming challenge that tests your problem-solving skills. In this project, you will build a Sudoku solver using Python and the backtracking algorithm. You will learn how to represent the puzzle, implement the solver, and visualize the solution using a graphical interface.

def is\_valid(board, row, col, num):  
 # Check if the number already exists in the row  
 for i in range(9):  
 if board[row][i] == num:  
 return False

# Check if the number already exists in the column  
 for i in range(9):  
 if board[i][col] == num:  
 return False # Check if the number already exists in the 3x3 grid  
 start\_row = (row // 3) \* 3  
 start\_col = (col // 3) \* 3  
 for i in range(3):  
 for j in range(3):  
 if board[start\_row + i][start\_col + j] == num:  
 return False return Truedef solve\_sudoku(board):  
 for row in range(9):  
 for col in range(9):  
 if board[row][col] == 0:  
 for num in range(1, 10):  
 if is\_valid(board, row, col, num):  
 board[row][col] = num if solve\_sudoku(board):  
 return True board[row][col] = 0 return False return Truedef print\_board(board):  
 for row in range(9):  
 for col in range(9):  
 print(board[row][col], end=" ")  
 print()# Example Sudoku board (0 represents empty cells)  
board = [  
 [5, 3, 0, 0, 7, 0, 0, 0, 0],  
 [6, 0, 0, 1, 9, 5, 0, 0, 0],  
 [0, 9, 8, 0, 0, 0, 0, 6, 0],  
 [8, 0, 0, 0, 6, 0, 0, 0, 3],  
 [4, 0, 0, 8, 0, 3, 0, 0, 1],  
 [7, 0, 0, 0, 2, 0, 0, 0, 6],  
 [0, 6, 0, 0, 0, 0, 2, 8, 0],  
 [0, 0, 0, 4, 1, 9, 0, 0, 5],  
 [0, 0, 0, 0, 8, 0, 0, 7, 9]  
]if solve\_sudoku(board):  
 print("Sudoku solved:")  
 print\_board(board)  
else:  
 print("No solution exists for the given Sudoku board.")

**6. Web Scraping with BeautifulSoup:**

Web scraping involves extracting data from websites, and it is a valuable skill in various domains. In this project, you will use Python and the BeautifulSoup library to scrape data from a website of your choice. You will learn how to navigate HTML structures, extract specific information, and save it to a file or a database.

import requests  
from bs4 import BeautifulSoup

# Send a GET request to the website  
url = 'https://example.com'  
response = requests.get(url)# Create a BeautifulSoup object  
soup = BeautifulSoup(response.text, 'html.parser')# Find and extract specific elements from the webpage  
title = soup.title.text  
paragraphs = soup.find\_all('p')# Print the extracted data  
print('Title:', title)  
print('Paragraphs:')  
for p in paragraphs:  
 print(p.text)

**7. Chatbot:**

Building a chatbot is an exciting project that combines natural language processing and machine learning. You can use Python and libraries like NLTK or spaCy to create a chatbot that can understand user queries and provide relevant responses. This project will introduce you to techniques such as text preprocessing, intent recognition, and response generation.

import random

# List of sample responses  
responses = [  
 "Hello!",  
 "Hi there!",  
 "Greetings!",  
 "Nice to meet you!",  
 "How can I assist you?",  
 "I'm here to help!",  
 "How are you today?",  
]def get\_random\_response():  
 """Return a random response from the list of sample responses."""  
 return random.choice(responses)def chat():  
 """Main function to handle the chatbot conversation."""  
 print("Chatbot: " + get\_random\_response()) while True:  
 user\_input = input("User: ")  
   
 # Check if the user wants to end the conversation  
 if user\_input.lower() == "bye":  
 print("Chatbot: Goodbye!")  
 break  
   
 # Generate and print a random response  
 print("Chatbot: " + get\_random\_response())if \_\_name\_\_ == "\_\_main\_\_":  
 print("Chatbot: Hello! How can I assist you?")  
 chat()

**8. Password Manager:**

A password manager is a useful tool for securely storing and managing passwords. In this project, you will develop a password manager using Python and cryptography libraries. Your program will allow users to store their passwords, generate strong passwords, and encrypt the data to ensure security.

import hashlib  
import getpass

passwords = {}def get\_hashed\_password(password):  
 """Generate a SHA-256 hashed password."""  
 sha256\_hash = hashlib.sha256()  
 sha256\_hash.update(password.encode('utf-8'))  
 return sha256\_hash.hexdigest()def create\_password():  
 """Create a new password entry."""  
 website = input("Enter the website: ")  
 username = input("Enter your username: ")  
 password = getpass.getpass("Enter your password: ")  
 hashed\_password = get\_hashed\_password(password)  
 passwords[website] = (username, hashed\_password)  
 print("Password created successfully.")def retrieve\_password():  
 """Retrieve a password from the password manager."""  
 website = input("Enter the website: ")  
 if website in passwords:  
 username, hashed\_password = passwords[website]  
 password = getpass.getpass("Enter your password: ")  
 if hashed\_password == get\_hashed\_password(password):  
 print(f"Username: {username}")  
 print(f"Password: {password}")  
 else:  
 print("Incorrect password.")  
 else:  
 print("Website not found in the password manager.")def main():  
 while True:  
 print("1. Create a new password")  
 print("2. Retrieve a password")  
 print("3. Quit")  
 choice = input("Enter your choice (1-3): ") if choice == "1":  
 create\_password()  
 elif choice == "2":  
 retrieve\_password()  
 elif choice == "3":  
 break  
 else:  
 print("Invalid choice. Please try again.")if \_\_name\_\_ == "\_\_main\_\_":  
 main()

**9. Stock Price Analyzer:**

Analyzing stock prices is crucial for investors and traders. In this project, you will create a stock price analyzer using Python and the Yahoo Finance API. You will fetch historical stock data, calculate various financial indicators, and visualize the results using charts. This project will enhance your data analysis and visualization skills.

import yfinance as yf  
import matplotlib.pyplot as plt

def analyze\_stock(symbol, start\_date, end\_date):  
 # Fetch the stock data from Yahoo Finance  
 stock\_data = yf.download(symbol, start=start\_date, end=end\_date) # Calculate the daily returns  
 stock\_data['Daily Return'] = stock\_data['Close'].pct\_change() # Plot the closing price and daily returns  
 plt.figure(figsize=(10, 5))  
 plt.subplot(2, 1, 1)  
 plt.plot(stock\_data['Close'])  
 plt.title('Stock Price')  
 plt.ylabel('Price') plt.subplot(2, 1, 2)  
 plt.plot(stock\_data['Daily Return'])  
 plt.title('Daily Returns')  
 plt.ylabel('Return') plt.tight\_layout()  
 plt.show()# Example usage  
symbol = 'AAPL' # Stock symbol (e.g., Apple Inc.)  
start\_date = '2022-01-01' # Start date of the analysis  
end\_date = '2022-12-31' # End date of the analysisanalyze\_stock(symbol, start\_date, end\_date)

**10. Automated Email Sender:**

Automating repetitive tasks is a common use case for Python. In this project, you will build an automated email sender that can send personalized emails to a list of recipients. You will use Python’s built-in email library to compose and send emails

programmatically. This project will provide insights into email protocols, handling attachments, and sending emails in bulk.

import smtplib  
from email.mime.multipart import MIMEMultipart  
from email.mime.text import MIMEText

def send\_email(sender\_email, sender\_password, recipient\_email, subject, message):  
 # Create a multipart message  
 msg = MIMEMultipart()  
 msg['From'] = sender\_email  
 msg['To'] = recipient\_email  
 msg['Subject'] = subject # Add the message body  
 msg.attach(MIMEText(message, 'plain')) # Setup the SMTP server  
 smtp\_server = 'smtp.gmail.com'  
 smtp\_port = 587 try:  
 # Start the SMTP server connection  
 server = smtplib.SMTP(smtp\_server, smtp\_port)  
 server.starttls() # Login to the email account  
 server.login(sender\_email, sender\_password) # Send the email  
 server.sendmail(sender\_email, recipient\_email, msg.as\_string()) print('Email sent successfully!')  
 except Exception as e:  
 print('An error occurred while sending the email:', str(e))  
 finally:  
 # Terminate the SMTP server connection  
 server.quit()# Example usage  
sender\_email = 'your-email@gmail.com' # Your Gmail email address  
sender\_password = 'your-password' # Your Gmail password  
recipient\_email = 'recipient-email@example.com' # Email address of the recipient  
subject = 'Automated Email' # Email subject  
message = 'Hello, this is an automated email.' # Email messagesend\_email(sender\_email, sender\_password, recipient\_email, subject, message)

**Conclusion**:  
Working on Python projects with code is an effective way to level up your programming skills. In this blog post, we explored ten diverse projects that cover areas like web development, data analysis, machine learning, and automation. By completing these projects, you will gain hands-on experience and develop a deeper understanding of Python and its libraries. So, pick a project that interests you, dive into the code, and unleash your creativity as you build practical applications with Python. Happy coding!