Mastering LLM Applications with LangChain and Hugging Face

Practical insights into LLM deployment and use cases

Hunaidkhan Pathan

Nayankumar Gajjar

www.bpbonline.com

First Edition 2025

Copyright © BPB Publications, India

ISBN: 978-93-65891-041

All Rights Reserved. No part of this publication may be reproduced, distributed or transmitted in any form or by any means or stored in a database or retrieval system, without the prior written permission of the publisher with the exception to the program listings which may be entered, stored and executed in a computer system, but they can not be reproduced by the means of publication, photocopy, recording, or by any electronic and mechanical means.

LIMITS OF LIABILITY AND DISCLAIMER OF WARRANTY

The information contained in this book is true to correct and the best of author's and publisher's knowledge. The author has made every effort to ensure the accuracy of these publications, but publisher cannot be held responsible for any loss or damage arising from any information in this book.

All trademarks referred to in the book are acknowledged as properties of their respective owners but BPB Publications cannot guarantee the accuracy of this information.

To View Complete
BPB Publications Catalogue
Scan the QR Code:

www.bpbonline.com

Dedicated to

I dedicate this book to my treasured parents, my beloved wife, my wonderful kids, and my esteemed mentor, ***Mr. Amit Saraswat****. Your unwavering support and guidance have been the cornerstone of my journey.*

– Hunaidkhan Pathan

Almighty, ***Dr. Amit Saraswat****, and My Family*

– Nayankumar Gajjar

About the Authors

- **Hunaidkhan Pathan** currently serves as a Data Science Lead for a leading consulting firm with over a decade of experience in the field. Specializing in machine learning and artificial intelligence, he brings a wealth of expertise to his role. Hunaidkhan holds a PGDM in Data Science from Shanti Business School in Ahmedabad and a degree in Electronics and Communication Engineering from Gujarat Technological University.

  He has significantly contributed to the data science community, with his research papers selected and presented at the prestigious SAS Analytics Conference 2013 in Orlando. The titles of his papers include "Marketing Mix Modeling" as an author and "Predicting market uncertainty with Kalman filter" as a co-author. He was also a LinkedIn Top Voice for Data Science and Artificial Intelligence in 2023. He posts regularly on LinkedIn about Generative AI.

  Hunaidkhan is an acknowledged **Subject Matter Expert** (**SME**) in Generative AI and Natural Language Processing. His diverse experience spans various LLM services such as OpenAI, Nvidia Nemo, Anthropic, GCP Generative AI, and AWS Bedrock, in addition to numerous open-source LLMs. His broad experience and profound knowledge make him a valuable contributor in the domain of data science.

- **Nayankumar Gajjar**, has a rich background in Data Science, Machine Learning and Generative AI fields with 9 years of extensive experience as a Data Scientist, Machine Learning Engineer, and Python Developer. Over the years, he has made significant contributions to various high-impact projects, showcasing his expertise in statistical modeling, Generative AI, MLOps, and Cloud Computing. This diverse skill set makes him a versatile and highly skilled professional in the Data Science and Machine Learning domains. He holds a master's degree in Decision Science, further solidifying his deep understanding of the field. In addition to his professional work, he is a YouTuber and a blogger who shares his experiences and knowledge, offering a complete understanding of statistics and providing detailed coding tutorials. His commitment to education extends to his role as a visiting faculty member, where he has taught Python, SQL, Data Science, and NLP courses. He also co-authored a research paper titled "Thiessen Polygon, A GIS approach for Retail Industry in SAS," which was presented at the prestigious SAS Analytics Conference 2013 in Orlando.

About the Reviewer

Vijender Singh is a multi-cloud professional with over six years of expertise, currently working in Luxembourg. He holds an MSc with distinction from Liverpool John Moores University, where his research centered on keyphrase extraction. Vijender boasts an impressive collection of cloud certifications, including Google MLPE, five Azure certifications, two AWS certifications, and TensorFlow certification. His role as a technical reviewer for numerous books reflects his commitment to improving the future.

Acknowledgements

We would like to express our sincere gratitude to all those who contributed to the completion of this book.

First and foremost, we extend our heartfelt appreciation to our mentor, Dr. Amit Saraswat, our family and friends for their unwavering support and encouragement throughout this journey. Their love and encouragement have been a constant source of motivation.

We are immensely grateful to BPB Publications for their guidance and expertise in bringing this book to fruition. Their support and assistance were invaluable in navigating the complexities of the publishing process.

We would also like to acknowledge the reviewers, technical experts, and editors who provided valuable feedback and contributed to the refinement of this manuscript. Their insights and suggestions have significantly enhanced the quality of the book.

Last but not least, we want to express our gratitude to the readers who have shown interest in our book. Your support and encouragement have been deeply appreciated.

Thank you to everyone who has played a part in making this book a reality.

Preface

In earlier days, when AI was in its beginning phase, we used to work with statical modeling, which contains statistical models like regression, random forest, decision tree, etc. At that time, we used to work with numerical data only, and we did not have much to gain from textual data. Gradually, we got a way under the umbrella of **Bag of Words** (**BoW**) through which we can work with textual data. The main logic was converting textual data to numerical data. For this, we have a few methods, like count vectors and TF-IDF vectors. These methods create a matrix that shows the occurrence of a word in the given document. Again, these methods were not helping ML models get the context or intent of what had been said in the text. These techniques were helping us to do sentiment analysis and other prediction-based tasks using the above mentioned algorithms.

Fast forward to this time, where we have some advanced techniques like transformers having an underlying architecture of neural networks, due to which ML models are able to get the context as well as the intent of what has been said in the text. This has opened up new opportunities and possibilities in the world of **Natural Language Processing** (**NLP**) and **Natural Language Generation** (**NLG**).

Both NLP and NLG are very important fields in the current era of AI. These fields give machines the power to understand and generate texts like human beings. Some of the readers must have heard the term "ChatGPT," one of the well-known chatbot platforms from OpenAI. If you have ever used ChatGPT, you must have an idea that it can write code for you, provide medical advice as well, do future prediction as well, and again, here you can chat with ChatGPT, similar to talking to a person and the person answering your questions.

As time passes, these text generation and understanding models become more advanced and able to perform and understand almost all text related tasks. To create such an advancement in the NLP and NLG areas, we will definitely need people who not only know but also have a better understanding of all the terminologies and concepts of NLP and NLG. Also, they should be aware of the steps and phases of the development and deployment of ML models to be served to end users. As we [authors] are interacting with different people in our day-to-day lives, we have found that there is no one step solution that can provide readers with all the above-mentioned things in one place. If readers get terminologies and concepts, then they will not get steps. If they get steps, then there is no practical exposure. If readers have practical exposure, then how to deploy on the cloud is another question. This book comes into the picture in such scenarios.

This book has been written for beginners or people who are stuck at the different stages mentioned in the previous paragraph and do not know about the next steps. This can be divided into three parts. In the first part, you can consider the first three chapters, where we have shown the installation of Python, running Python scripts in different ways, the basic concepts of Python, the installation of editors, and the usage and importance of the virtual environment. In the second part, you can consider chapters 4 and 5, which show the basic and important concepts of NLP and NLG. From chapters 6 to 11, we have shown the usage of important packages like LangChain and Hugging Face. Then we have shown how you can create a chatbot with custom data and integrate it with an application like Telegram. At last, we have shown deployment to an AWS cloud environment. The rest of the chapters are related to future direction and include some useful tips and references.

In this book, we have not only discussed the theoretical approach, but we have also implemented and provided practical exposure as well. In the practical implementation, you will learn all the required steps to be performed to make things work.

We hope that this book will be helpful to any individual who is looking forward to starting their journey in the NLP and NLG fields. We also hope that this book will provide complete guidance and help readers to the required understanding with practical exposure.

Chapter 1: Introduction to Python and Code Editors – In this chapter, readers will learn about Python as a programming language and its history. Readers will get an idea of Python's features and why it is an important language from an AI/ML perspective. Also, the reader will get an idea about the difference between a code editor and an **Integrated Development Environment (IDE)**.

Chapter 2: Installation of Python, Required Packages, and Code Editors – In this chapter, readers will install Python, all the packages we are going to use throughout the entire book, and an IDE to start with coding. Apart from the installation, readers will gain knowledge on the virtual environment, its importance and its usage. Also, readers will gain knowledge and practical exposure to Python programming basics.

Chapter 3: Ways to Run Python Scripts – In this chapter, readers will create their first Python script, and then they will get practical hands-on experience on different ways to run any Python script.

Chapter 4: Introduction of NLP and its concepts – In this chapter, readers will get exposure to the theoretical concepts and terminologies of NLP, which are essential to start with. Also, readers will get practical hands-on experience with all the important terminologies and concepts.

Chapter 5: Introduction to Large Language Models – This chapter contains theoretical concepts. In this chapter, readers will acquire knowledge on LLM history and its evaluation. Apart from the history, readers will also learn important terminologies and concepts of LLMs.

Chapter 6: Introduction to LangChain, Usage and Importance – In this chapter, readers will gain knowledge of the LangChain package, which is mainly used for text data **Extract, Transform, Load** (**ETL**) tasks to be later used by LLMs for further processing, understanding, and text generation. Readers will get to know LangChain integration with Hugging Face and how to use LLMs available from Hugging Face. In the chapter, readers will also get practical exposure, which will help them practice and gain confidence.

Chapter 7: Introduction to Hugging Face, its Usage and Importance – In this chapter, readers will get practical exposure to the different LLMs available on Hugging Face Hub and how to use them. Readers will explore Hugging Face Hub as well, which provides a complete ecosystem for LLM deployment.

Chapter 8: Creating Chatbots using Custom Data with Langchain and Hugging Face Hub – In this chapter, readers will create chatbots using the RAG mechanism on custom data using LangChain and Hugging Face combinations. Also, readers will get exposure to the Gradio framework of Hugging Face, through which they can interact with the chatbot created.

Chapter 9: Hyperparameter Tuning and Fine Tuning Pre-Trained Models – In this chapter, the user will gain knowledge about the different hyperparameters available for any LLM, their usage, and how they will impact the LLM's performance.

Chapter 10: Integrating LLMs into Real-World Applications: Case Studies – In this chapter, readers will create a Telegram chatbot with the custom data and interact with it. Readers will get step-by-step guide on the implementation.

Chapter 11: Deploying LLMs in Cloud Environments for Scalability – In this chapter, readers will get a step-by-step guide to deploying chatbots and LLM models in an AWS cloud environment. Readers will also get an idea about GCP.

Chapter 12: Future Directions: Advances in LLMs and Beyond – In this chapter, readers will learn future directions and where to go from here once the book has been completed.

Appendix A: Useful Tips for Efficient LLM Experimentation – In this chapter, we have shared some tips to use LLMs more efficiently.

Appendix B: Resources and References – In this chapter, we have provided some of the resources and references for the readers to get more depth and detailed knowledge on different models and packages.

Code Bundle and Coloured Images

Please follow the link to download the
Code Bundle and the ***Coloured Images*** of the book:

https://rebrand.ly/bf9408

The code bundle for the book is also hosted on GitHub at
https://github.com/bpbpublications/Mastering-LLM-Applications-with-LangChain-and-Hugging-Face.
In case there's an update to the code, it will be updated on the existing GitHub repository.

We have code bundles from our rich catalogue of books and videos available at **https://github.com/bpbpublications**. Check them out!

Errata

We take immense pride in our work at BPB Publications and follow best practices to ensure the accuracy of our content to provide with an indulging reading experience to our subscribers. Our readers are our mirrors, and we use their inputs to reflect and improve upon human errors, if any, that may have occurred during the publishing processes involved. To let us maintain the quality and help us reach out to any readers who might be having difficulties due to any unforeseen errors, please write to us at :

errata@bpbonline.com

Your support, suggestions and feedbacks are highly appreciated by the BPB Publications' Family.

Did you know that BPB offers eBook versions of every book published, with PDF and ePub files available? You can upgrade to the eBook version at www.bpbonline.com and as a print book customer, you are entitled to a discount on the eBook copy. Get in touch with us at :

business@bpbonline.com for more details.

At **www.bpbonline.com**, you can also read a collection of free technical articles, sign up for a range of free newsletters, and receive exclusive discounts and offers on BPB books and eBooks.

Piracy

If you come across any illegal copies of our works in any form on the internet, we would be grateful if you would provide us with the location address or website name. Please contact us at **business@bpbonline.com** with a link to the material.

If you are interested in becoming an author

If there is a topic that you have expertise in, and you are interested in either writing or contributing to a book, please visit **www.bpbonline.com**. We have worked with thousands of developers and tech professionals, just like you, to help them share their insights with the global tech community. You can make a general application, apply for a specific hot topic that we are recruiting an author for, or submit your own idea.

Reviews

Please leave a review. Once you have read and used this book, why not leave a review on the site that you purchased it from? Potential readers can then see and use your unbiased opinion to make purchase decisions. We at BPB can understand what you think about our products, and our authors can see your feedback on their book. Thank you!

For more information about BPB, please visit **www.bpbonline.com**.

Join our book's Discord space

Join the book's Discord Workspace for Latest updates, Offers, Tech happenings around the world, New Release and Sessions with the Authors:

https://discord.bpbonline.com

Table of Contents

1. Introduction to Python and Code Editors 1

Introduction 1

Structure 1

Objectives 1

Introduction to Python 2

Introduction to code editors 5

Conclusion 6

References 7

Further reading 7

2. Installation of Python, Required Packages, and Code Editors 9

Introduction 9

Structure 9

Objectives 10

General instructions 10

Installation of Python on Windows 11

Installation of Python on Linux 13

Installation of Python on MacOS 13

Using Docker for Python 14

Installation of IDE 15

Installation of PyCharm 16

Installation of required packages 17

Virtual environment 17

virtualenv 18

pipenv 19

Folder structure 19

Creating a virtual environment 21

PEP 8 standards 23

Following PEP 8 in PyCharm *25*
Object-Oriented Programming concepts in Python 26
Classes in Python *29*
Functions in Python *32*
For loop in Python *33*
While loop in Python *34*
If-else in Python *35*
Conclusion 36

3. Ways to Run Python Scripts 37

Introduction 37
Structure 37
Objectives 38
Setting up the project 38
Running Python scripts from PyCharm 42
Running Python Scripts from Terminal 47
Running Python scripts from Jupyter Lab and Notebook 49
Running Python Scripts from Docker 52
Conclusion 54

4. Introduction to NLP and its Concepts 55

Introduction 55
Structure 55
Objectives 56
Natural Language Processing overview 56
Key concepts *56*
Corpus *57*
N-grams *59*
Tokenization *63*
Difference in tokens and n-grams *67*
Stop words removal *67*
Stemming *70*

Lemmatization ... *70*
Lowercasing ... *74*
Part-of-speech tagging ... *74*
Named Entity Recognition ... *76*
Bag of words ... *79*
Word embeddings ... *83*
Topic modeling ... *86*
Sentiment analysis ... *88*
Large language models ... 90
Transfer learning ... *91*
Text classification ... 91
Prompt engineering ... 96
Hallucination ... *96*
Syntactic relationship ... *96*
Semantic relationship ... *96*
Conclusion ... 97

5. Introduction to Large Language Models ... 99

Introduction ... 99
Structure ... 99
Objectives ... 100
History ... 100
LLM use cases ... 102
LLM terminologies ... 104
Neural networks ... 111
Transformers ... 116
Pre-built transformers ... 127
Bidirectional Encoder Representations from Transformers ... *128*
Generative Pre-trained Transformer ... *129*
Text-to-text transfer transformer ... *130*
DistilBERT ... *130*
XLNet ... *131*

RoBERTa *132*
Conclusion 132
Further readings 133
References 134

6. Introduction of LangChain, Usage and Importance 135

Introduction 135
Structure 135
Objectives 136
LangChain overview 136
Installation and setup 137
Usages 140
Opensource LLM models usage 140
Data loaders 151
Opensource text embedding models usage 153
Vector stores 162
Model comparison 169
Evaluation 175
Types of evaluation *175*
Conclusion 194
Points to remember 195
References 195

7. Introduction of Hugging Face, its Usage and Importance 197

Introduction 197
Structure 197
Objectives 198
Exploring the Hugging Face platform 198
Installation and setup 203
Datasets 203
Usage of opensource LLMs 213
Generating vector embeddings 222
Evaluation 225

Transfer learning with Hugging Face API 232
Real-world use cases of Hugging Face 234
Conclusion 234
References 235

8. Creating Chatbots Using Custom Data with LangChain and Hugging Face Hub 237

Introduction 237
Structure 238
Objectives 238
Setup 238
Overview 238
Steps to create RAG based chatbot with custom data 242
Dolly-V2-3B details 255
Data loaders by LangChain 255
Vector stores by LangChain 256
Conclusion 256
References 257

9. Hyperparameter Tuning and Fine Tuning Pre-Trained Models 259

Introduction 259
Structure 260
Objectives 260
Hyperparameters of an LLM 260
Hyperparameters at inferencing or at text generation 263
Fine-tuning of an LLM 267
Data preparation for finetuning an LLM 267
Performance improvement 275
Conclusion 275
References 276

10. Integrating LLMs into Real-World Applications – Case Studies 277

Introduction 277
Structure 277

Objectives 278
Case studies 278
Use case with Telegram 280
Setup *281*
Conclusion 289
References 290

11. Deploying LLMs in Cloud Environments for Scalability 291

Introduction 291
Structure 292
Objectives 292
Amazon Web Services 292
Step 1: Creating an Amazon SageMaker Notebook Instance *293*
Step 2: Create folders in SageMaker to store data *297*
Step 3: Create vector embeddings *299*
Step 4: Auto scaling *309*
Google Cloud Platform 311
Conclusion 311
References 312

12. Future Directions: Advances in LLMs and Beyond 313

Introduction 313
Structure 313
Objectives 314
Generative AI market growth 314
Reasoning 314
Emergence of multimodal models 315
Small domain-specific models 315
Multi agent framework *317*
Quantization and Parameter-Efficient Fine Tuning 317
Vector databases 318
Guardrails 319
Model evaluation frameworks 320

Ethical and bias mitigation 324
Safety and security 325
Conclusion 326
References 327

Appendix A: Useful Tips for Efficient LLM Experimentation 329

Structure 329
Objectives 330
Understanding the challenges of LLM experimentation 330
Preparing data for LLM experimentation 332
Optimizing model architecture and hyperparameters 333
Efficient training strategies for LLMs 335
Evaluating and interpreting experimental results 336
Fine-tuning for specific applications 338
Scaling up: Distributed training and parallel processing 339
Deployment considerations for LLMs 341
Conclusion 343
References 343

Appendix B: Resources and References 345

Introduction 345
Books and articles 345
Research papers 346
LangChain resources 347
Hugging Face resources 347
Alternative resources to LangChain 348
Community and support 350
Other important resources 350
Conclusion 350

Index 351-356

Chapter 1
Introduction to Python and Code Editors

Introduction

Python is a really powerful programming language that is simple and easy to read. This language is being used in many technology areas. There are rules called **Python Enhancement Proposal** (**PEP**) standards, which help you write proper Python code. These rules give instructions for how we should write and develop the programming language which helps us keep our code clean and of high quality!

There are also different ways that we can work on our Python codes - either through code editors or **Integrated Development Environments** (**IDEs**).

Structure

In this chapter we will discuss the following topics:

- Introduction to Python
- Introduction to code editors

Objectives

Learning Python well is a great start for getting into generative AI. Python is known for being easy to understand and has lots of tools you can use. It is a good language to learn

if you want to understand how programming works. Since Python is used everywhere, it is important for learning different kinds of machine learning, which is really helpful if you are interested in generative AI. If you are good at Python, you can easily use the important tools and information you need for generative AI. This makes it easier to move on to more advanced things like understanding how computers understand human language or deep learning.

Introduction to Python

Created by *Guido Van Rossum* back in the late 1980s, today, almost everybody uses Python! Here is a brief introduction to Python:

- **Readability**: It is easy to understand any code written by another person due to its simplicity.
- **Interpreted language**: Python is an interpreted language. You do not need to compile your program before running it on the system. This makes development faster and easier, as you can execute code line by line, and easy to debug.
- **Cross-platform**: Your computer runs Windows? MacOS? Linux? Do not worry! No matter what operating system your computer has installed, they all support Python!
- **Versatility**: Many more useful features like versatility, object-oriented libraries, and interfaces, community support, dynamic variable declaration, etc., which make work super smooth.
- **Libraries and frameworks**: If asked to paint an image without a canvas color brush, it is hard for anyone. In similar cases with computer programming, we require a lot of tools to make it happen; IDEs and code editors are one of them, which have unique functionality based on requirements and can be customized accordingly. For example, PyCharm is a Python-specific editor. We also have something called Jupyter Notebook or Jupyter Lab in the Python world. Python provides a large collection of different libraries for different tasks. For example, Django is used for web development, NumPy and Pandas are used for data analysis, and Tensorflow and Keras are used for deep learning.
- **Community and support**: Python's simplicity and community support make it a highly usable coding language. It does not matter if you are new to programming or an experienced pro: Python has something for everyone! This powerful tool will always come in handy whether it is web development, data analysis or AI/ML tasks etc.
- **Open source**: Python is open-source and free to use. This encourages collaboration and innovation, as anyone can contribute to the language's development or create their own Python packages.
- **Object-oriented**: Python is an object-oriented programming language, which means code organization around objects/classes simplifies managing complex systems.

- **Dynamic typing**: Python uses dynamic typing, which means variables do not require any explicit type definition speeding up development but further necessitates attention towards avoiding potential typing errors.
- **High-level language**: Python is a high-level programming language; since lower-level complexities are abstracted away, users can concentrate mainly on problem-solving and worrying less about underlying hardware details.
- **Duck typing**: Python follows the principle of *duck typing*, which means the object type determination is based on its behavior which gives code the freedom to be concise yet vigilant towards object compatibility.
- **Multi-paradigm**: Python supports multiple programming instances, including procedural, object-oriented, and functional programming. This versatility allows us to choose the most suitable approach for our project's requirements.
- **Interoperability**: Python interacts smoothly with other languages like C, C++, Java, etc., enabling utilization of existing libraries/code.
- **Popular use cases**: Usable across several domains, i.e., web development (using Django/Flask/FastAPI), data science (machine learning using Scikit-Learn/TensorFlow), Scientific computing (using NumPy/SciPy), automation scripting or even game development, the list is endless.
- **Python 2 vs. Python 3**: It is important to note that there are two major versions of Python: Python 2 and Python 3. Though there are mainly two versions available, as of Jan 1st, 2020, only Python 3 receives updates/supports - have an edge by starting all new projects in this version!

In conclusion, regardless of whether you are a beginner embarking upon an initial language learning journey or an experienced developer handling intricate setup, the simplicity/readability/use case versatility/community backup is positioning Python as handy across several coding tasks.

Zen (**Python Enhancement Proposal PEP 20)** Philosophy embraces design ideals/principles defining how Python code should be written for not just computers but easy understanding by fellow developers too!

Here are some of the key principles from the *Zen of Python* written by *Tim Peters*:

- **Beautiful is better than ugly**: Python code should be aesthetically pleasing, clear, and elegant. This encourages developers to write code that is not only functional but also visually appealing.
- **Explicit is better than implicit**: Code should be explicit in its intentions and behavior. Avoid relying on hidden or implicit features to make the code more understandable.
- **Simple is better than complex**: Simplicity is preferred over complexity. The code should be straightforward and easy to understand rather than unnecessarily convoluted.

- **Complex is better than complicated**: While simplicity is encouraged and complexity is necessary, it should be well-structured and not overly complicated. Complex code should have a clear purpose and design.
- **Flat is better than nested**: Deeply nested code structures should be avoided. Keeping code relatively flat, with fewer levels of indentation, makes it more readable and maintainable.
- **Sparse is better than dense**: Code should be spaced out and not overly dense. Proper spacing and indentation enhance readability.
- **Readability counts**: Readability is a top priority in Python. Code should be written with the goal of making it easy to read and understand, not just for the computer but also for other developers.
- **Special cases are insufficient to break the rules**: Consistency is important. While there may be exceptional cases, they should not lead to a violation of established coding conventions and rules.
- **Practicality beats purity**: While adhering to best practices and principles is important, practicality should not be sacrificed in the pursuit of theoretical perfection. Real-world solutions sometimes require pragmatic compromises.
- **Errors should never pass silently. Unless explicitly silenced**: Errors and exceptions should be handled explicitly. If you encounter an error, it should not be ignored or suppressed unless you have a good reason to do so.
- **In the face of ambiguity, refuse the temptation to guess**: When faced with uncertainty or ambiguity in your code, it is better to be explicit and not make assumptions. Clarity should prevail.
- **There should be one and preferably only one obvious way to do it**: Python encourages a single, clear way to accomplish tasks to minimize confusion and inconsistency in code.
- **Although that way may not be obvious at first unless you are Dutch**: This lighthearted remark acknowledges that not all design decisions may immediately make sense to everyone and hints at Python's creator, *Guido van Rossum*.
- **Now is better than never. Although never is often better than right now**: While taking action is important, rushing without proper consideration can lead to errors. It's a reminder to balance speed with careful thought.
- **If the implementation is hard to explain, it is a bad idea. If the implementation is easy to explain, it may be a good idea**: Code should be designed in a way that makes its purpose and behavior clear and straightforward. Complex, hard-to-explain implementations should be avoided.
- **Namespaces are one honking great idea: Let us do more of those**: Encouragement to use namespaces for organizing and managing variables and functions, promoting modularity, and avoiding naming conflicts.