e-Book

PLAY WITH PYTHON

- MR. PAWAN KUMAR
- DR. ANUPA SINHA
- MRS. AKANKSHA MISHRA
- MRS. SUSHREE SASMITA DASH.

Play with

PYTHON

By

Mr. Pawan Kumar[*1]
Dr. Anupa Sinha[2]
Mrs. Akanksha Mishra[3]
Mrs. Sushree Sasmita Dash[4]

***1, 2, 3, 4 Assistant Professor,**
Department of CS & IT , Kalinga University, Kotni, Raipur, Chhattisgarh, 492001

Published by:

ISBN: 978-93-340-0763-3

Date: 25-12-2023
Innovation and Integrative Research Center Journal
ISSN: 2584-1491.
Address:- Kota, Raipur, Chhattisgarh, India.
492001.
Contact No.- +91-8871386512
Email: connect@samagracs.com

PREFACE

Welcome to "Play with Python"! This book is designed to be your companion on an exciting journey through the world of programming using Python. Whether you're a complete novice or an experienced coder looking to expand your skills, this book offers something for everyone. Python is not just a programming language; it's a powerful tool that can unlock a world of creativity, problem-solving, and innovation. Its simplicity and readability make it an ideal choice for beginners, while its versatility and vast ecosystem of libraries make it a favourite among professionals across various fields.

In "Play with Python," we'll take a hands-on approach to learning. Instead of drowning you in theory and syntax, we'll dive right into practical examples, projects, and exercises that will have you coding from day one. Each chapter builds upon the previous one, gradually introducing new concepts and techniques while reinforcing what you've already learned. Whether you dream of building your own web applications, analysing data to uncover insights, automating repetitive tasks, or even delving into artificial intelligence and machine learning, Python can take you there. And this book will be your guide along the way.

But remember, learning to code is like learning any other skill—it takes time, practice, and patience. Don't be discouraged by setbacks or challenges. Embrace them as opportunities to grow and improve. And always remember to have fun! After all, that's what "Play with Python" is all about. So, grab your keyboard, fire up your interpreter, and let's play with Python!

Warm regards,
Mr. Pawan Kumar,
Dr. Anupa Sinha,
Mrs. Akanksha Mishra,
Mrs. Sushree Sasmita Dash.

Table of Content

Serial Number	Title	Page Number
1	Introduction to Python	1
2	Setting up Python	6
3	Variables and Data Types	13
4	Basic Operations	20
5	Control Flow	29
6	Data Structures	38
7	Functions	50
8	File Handling	56
9	Exception Handling	63
10	Object-Oriented Programming (OOP)	68
11	Libraries and Modules	78
12	Introduction to NumPy and Pandas	86
13	Basic Input/Output	98
14	Basic Concepts of Web Development	102
15	Graphics Design in Python	109

1. Introduction to Python

Python, a dynamic and versatile programming language, has rapidly emerged as a cornerstone in the world of software development. Created by Guido van Ros

Introduction to Python:

Python was created by Guido van Rossum, a Dutch programmer, in the late 1980s. Guido started working on Python in December 1989 and released its first version, Python 0.9.0, in February 1991. He envisioned a programming language that prioritized readability, simplicity, and developer productivity. Over the years, Python has grown to become one of the most popular and versatile programming languages, known for its clean syntax and extensive standard library.

Guido van Rossum served as the "Benevolent Dictator For Life" (BDFL) of the Python community until he stepped down from the role in July 2018. Despite his step down, Guido remains a respected figure in the Python world, and his contributions have profoundly influenced the development and success of Python as a programming language.

Overview of Python's history and development.

Python's history and development span several decades, marked by steady growth, community engagement, and adaptability. Here's an overview of key milestones in Python's journey:

1. **Inception (1989-1991)**: Python was conceived by Guido van Rossum, a Dutch programmer, in the late 1980s. Van Rossum aimed to create a language that combined the benefits of ABC and Modula-3, focusing on simplicity and readability. The first official Python release, Python 0.9.0, came in February 1991.
2. **Python 1.0 (January 1994)**: The release of Python 1.0 marked a significant milestone, solidifying Python's position as a capable and versatile programming language. During this time, the community began to form around the language, contributing to its early growth.
3. **Python 2.0 and the Python Software Foundation (2000):** Python 2.0, released in October 2000, introduced list comprehensions, garbage

collection, and Unicode support. Around the same time, the Python Software Foundation (PSF) was established as a non-profit organization to promote, protect, and advance Python. This period saw increased collaboration and community involvement.

4. **Python 3.0 (December 2008**): Python 3.0, also known as Python 3000 or simply Py3k, marked a major shift in the language. It aimed to rectify design flaws and introduce new features, which led to some backward incompatibilities with Python 2. The decision to transition from Python 2 to Python 3 sparked discussions within the community and initiated a gradual migration process.
5. **Python 2 End of Life (January 1, 2020)**: With the community's encouragement, Python 2 reached its end of life on January 1, 2020. This milestone emphasized the importance of transitioning to Python 3, ensuring that the community focused on the latest and most supported version.
6. **Continuous Development (2020s)**: Python's development has continued with regular releases and enhancements. The language remains at the forefront of technological advancements, with a strong focus on data science, machine learning, web development, and more. The community, guided by the Python Enhancement Proposal (PEP) process, actively contributes to the language's growth and refinement.
7. **Current State (2024)**: As of my last knowledge update in January 2022, Python continued to be one of the most widely used programming languages globally. Its popularity in various domains, including web development, data science, artificial intelligence, and automation, showcased its adaptability and relevance.

The language's evolution and robust community support make it a staple in the toolkit of developers around the world.

Python's journey reflects its commitment to simplicity, readability, and adaptability, making it a language that appeals to both beginners and seasoned developers across diverse domains.

Python's popularity and use cases.

It seems there might be a slight confusion in your question. I'll provide information on both IPython and Python, as they are related but distinct concepts.

1. Python:

Python is a high-level, general-purpose programming language known for its readability, simplicity, and versatility. Its popularity has grown significantly over the years, making it one of the most widely used programming languages. Python is employed in various domains, including:

- **Web Development**: Frameworks like Django and Flask enable developers to build scalable and robust web applications.
- **Data Science**: Libraries like NumPy, pandas, and scikit-learn facilitate data manipulation, analysis, and machine learning.
- **Artificial Intelligence**: Python has become a primary language for AI and machine learning projects, thanks to frameworks like TensorFlow and PyTorch.
- **Automation and Scripting**: Python's simplicity makes it ideal for automating repetitive tasks and writing scripts for various purposes.
- Scientific Computing: Python is used in scientific research and engineering for tasks involving numerical and scientific computing.

2. IPython:

IPython, on the other hand, is an interactive command-line shell for Python. It provides an enhanced Read-Eval-Print Loop (REPL) with additional features compared to the standard Python shell. Some key features of IPython include:

- **Interactive Computing:** IPython allows users to interactively explore data, run code snippets, and experiment with Python syntax in a more user-friendly environment than the default Python interpreter.
- **Rich Output**: IPython supports rich media output, including images, videos, and interactive plots, enhancing the overall interactive experience.
- **History and Command Editing**: Users can access command history, edit previous commands, and leverage various keyboard shortcuts for improved productivity.
- **Support for Parallel Computing**: IPython supports parallel and distributed computing, enabling users to harness the power of multicore processors or distributed computing environments.

IPython is often used by data scientists, researchers, and educators who require an interactive and exploratory environment for working with Python. Its features make it particularly well-suited for tasks like data analysis, scientific research, and educational purposes.

In summary, while Python serves as a general-purpose programming language applicable to a wide range of domains, IPython complements Python by offering an enhanced interactive environment tailored for exploration, experimentation, and data-related tasks. Both Python and IPython contribute to the overall ecosystem of tools available to developers and researchers.

Comparison with other programming languages.

Comparing programming languages can be complex and depends on various factors, including the context of usage, project requirements, developer preferences, and specific features of each language. Here's a broad comparison of Python with some popular programming languages:

Python vs. Java:

- Syntax: Python's syntax is generally considered more concise and readable than Java's.
- Performance: Java is often perceived as more performant than Python, especially in large-scale enterprise applications.
- Use Cases: Python is favored for data science, machine learning, and scripting, while Java is widely used for building enterprise-level applications.

Python vs. JavaScript:

- Context of Use: Python is often used on the server side (Django, Flask), in data science, and scripting. JavaScript is the primary language for web development, both on the client side (browser) and server side (Node.js).
- Asynchronous Programming: JavaScript is inherently asynchronous, making it well-suited for handling concurrent operations on the web.

Python vs. C++:

- Performance: C++ is generally faster than Python as it is a compiled language, while Python is an interpreted language.
- Memory Management: C++ provides manual memory management, giving developers more control, whereas Python has automatic memory management through garbage collection.
- Use Cases: C++ is often used for system-level programming, game development, and performance-critical applications, while Python is preferred for its ease of use and readability.

Python vs. Ruby:

- Philosophy: Both Python and Ruby prioritize readability and developer happiness, but Python has a stronger emphasis on simplicity.
- Community and Ecosystem: Python has a larger community and a broader ecosystem, especially in scientific computing and machine learning.
- Web Development: Ruby is well-known for the Ruby on Rails web framework, while Python has frameworks like Django and Flask.

Python vs. C#:

- Platform: C# is primarily associated with the Microsoft .NET platform, while Python is more platform-agnostic.
- Use Cases: C# is often used for Windows desktop applications, game development with Unity, and enterprise-level applications. Python has strengths in web development, data science, and artificial intelligence.

Python vs. Go:

- Concurrency: Go is designed with concurrency in mind, making it well-**suited for building scalable and concurrent systems. Python also supports concurrency but in a different manner.
- Performance: Go tends to have better performance in terms of raw speed and efficiency for certain use cases.
- Use Cases: Go is commonly used for cloud services, networking, and containerization (e.g., Kubernetes), while Python is versatile across various domains.

2. Setting up Python

Installing Python can be done using various methods, and two commonly used approaches are via the official Python website and the Anaconda distribution. Here's a guide for each method:

Installing Python from the Official Website:

- Visit the Python Official Website:
- Go to the Python official website.
- **Download Python:**
- Click on the "Downloads" tab.
- You'll see the latest version of Python. If you're starting, it's recommended to download the latest stable version.
- Choose the installer based on your operating system (Windows, macOS, or Linux).

Run the Installer:

- For Windows: Double-click the downloaded .exe file and follow the installation wizard.
- For macOS: Double-click the downloaded .pkg file and follow the installation instructions.
- For Linux: Open a terminal, navigate to the downloaded file's directory, and run the installation command (replace X.Y with the Python version):

```
sudo apt-get update

sudo apt-get install pythonX.Y

Verify Installation:

Open a command prompt or terminal.

Type python --version or python3 --version and press Enter to ensure Python is installed.
```

Installing Python using Anaconda:

Anaconda is a distribution that includes Python, popular libraries, and tools for data science.

Download Anaconda:

- Visit the Anaconda download page.
- Choose the appropriate version for your operating system (Windows, macOS, or Linux).

Run the Installer:

- For Windows: Double-click the downloaded .exe file and follow the installation wizard.
- For macOS: Double-click the downloaded .pkg file and follow the installation instructions.
- For Linux: Open a terminal, navigate to the downloaded file's directory, and run the installation command:

```
bash Anaconda3-X.Y.Z-Linux-x86_64.sh
```

Follow the prompts to complete the installation.

Verify Installation:

- Open a new terminal or command prompt.
- Type `conda --version` to verify that the Anaconda package manager is installed.
- Type `python --version` to check the Python version.

Update Anaconda:

- It's a good practice to regularly update Anaconda. In the terminal or command prompt, run:

```
conda update --all
```

These methods provide a straightforward way to install Python on your system. The choice between the official website and Anaconda depends on your specific needs. Anaconda is particularly useful for data science projects, as it comes with pre-installed libraries like NumPy, pandas, and Jupyter Notebook.

Choosing and setting up a code editor.

Choosing and setting up a code editor is an important step in the development process, as it significantly influences your coding experience. Here's a guide to help you choose and set up a code editor for Python development:

1. Choosing a Code Editor:

a. Visual Studio Code (VSCode):

Pros:

- Lightweight and fast.
- Excellent Python support with extensions.
- Rich ecosystem with a variety of extensions for different languages.
- Integrated terminal and debugging tools.

Cons:

- Requires some configuration for optimal Python support.

b. PyCharm:

Pros:

- Robust IDE specifically designed for Python.
- Smart code completion and navigation.
- Integrated testing and debugging tools.

Cons:

- Heavier compared to lightweight editors.

c. Atom:

Pros:

- Highly customizable.

- Large library of community-contributed packages.

Cons:

- May require additional packages for optimal Python development.

d. Sublime Text:

Pros:

- Extremely fast and lightweight.
- Large selection of plugins and themes.

Cons:

- Less integrated Python support compared to some IDEs.

2. Setting Up Visual Studio Code (VSCode) as an Example:

a. Download and Install:

- Download VSCode from the official website.
- Follow the installation instructions for your operating system.

b. Python Extension:

- Open VSCode and go to the Extensions view by clicking on the Extensions icon in the Activity Bar on the side of the window.
- Search for "Python" and install the one published by Microsoft.
- This extension provides language support, debugging, linting, IntelliSense, and more.

c. Setting Up Virtual Environment:

- Open a terminal in VSCode.
- Create a virtual environment using the following commands:

```
python -m venv venv
```

- This creates a virtual environment named 'venv'.
- Activate the virtual environment:

For Windows:

```
.\venv\Scripts\activate
```

For macOS/Linux:

```
source venv/bin/activate
```

d. Configure Python Interpreter:

In the bottom-right corner of VSCode, click on the interpreter version (it will show "Select Python Interpreter").

Choose the interpreter from the virtual environment you just created.

e. Install Additional Extensions (Optional):

Explore and install additional extensions based on your preferences, such as Git integration, themes, or linting tools.

f. Start Coding:

- Create a new Python file (e.g., main.py).
- Write your Python code and run it from within VSCode.

3. Additional Tips:

Customization: Explore VSCode settings (Ctrl + ,) to customize the editor according to your preferences.

Install themes and icons from the Visual Studio Code marketplace to personalize your coding environment.

Configuring the development environment.

Configuring a development environment for Python involves setting up various tools, dependencies, and configurations to ensure a smooth and productive coding experience. Here's a comprehensive guide to help you configure your Python development environment:

1. Install Python:

Download and install the latest version of Python from the official Python website or use a package manager like apt (for Linux) or Homebrew (for macOS).

2. Setting Up a Virtual Environment:

- Use virtual environments to isolate project dependencies.
- Open a terminal and navigate to your project directory.

- Create a virtual environment:
  python -m venv venv

Activate the virtual environment:

For Windows:

```
.\venv\Scripts\activate
```

For macOS/Linux:

```
source venv/bin/activate
```

3. Install a Code Editor:

- Choose a code editor based on your preferences (e.g., Visual Studio Code, PyCharm, Atom, Sublime Text).
- Follow the installation and setup instructions for your chosen editor.

4. Configure Code Editor for Python:

- Install Python extensions for your editor (e.g., Python extension for Visual Studio Code).
- Configure the Python interpreter in your editor to use the virtual environment you created.

5. Version Control (Optional):

- Install Git for version control.
- Configure Git with your username and email:

```
git config --global user.name "Your Name"
git config --global user.email "your.email@example.com"
```

6. Package Management:

- Familiarize yourself with Python package management tools.
- Use pip to install project dependencies:

```
pip install package-name
```

7. Useful Python Packages: Install common Python packages for development:

```
pip install numpy pandas matplotlib requests
```

8. Database Setup (Optional): Install and configure any databases you plan to use (e.g., SQLite, PostgreSQL, MySQL).

9. Testing Setup (Optional): Install testing frameworks like pytest:

```
pip install pytest
```

10. Configure Environment Variables:

- Manage sensitive information or configuration settings using environment variables.
- Consider using tools like python-dotenv for loading environment variables from a .env file.

11. Explore Editor Settings and Extensions:

- Customize your code editor with themes, extensions, and settings that enhance your workflow.
- Familiarize yourself with keyboard shortcuts for efficient coding.

3. Variables and Data Types

In Python, declaring and initializing variables is a straightforward process. Unlike some other programming languages, Python is dynamically typed, which means you don't need to explicitly declare the data type of a variable. Here's how you can declare and initialize variables in Python:

1. Basic Variable Assignment:

You can assign values to variables in a single line, and Python will determine the data type dynamically.

```
# Integer variable
age = 25
# Float variable
height = 5.9
# String variable
name = "John Doe"
# Boolean variable
is_student = True
```

2. Multiple Assignments:

You can assign values to multiple variables in a single line.

```
# Multiple assignments
x, y, z = 5, 10, 15
```

3. Reassigning Variables:

You can change the value of a variable by assigning it a new value.

```
# Reassigning a variable
count = 10
count = count + 1  # Incrementing the value
```

4. Dynamic Typing:

Python is dynamically typed, so you can change the type of a variable during runtime.python

```
# Dynamic typing
dynamic_var = 42
print(dynamic_var)  # Output: 42

dynamic_var = "Hello"
print(dynamic_var)  # Output: Hello
```

5. None Type:

In Python, you can use the None type to represent the absence of a value or a null value.

```
# None type
result = None
```

6. Type Annotations (Optional):

Although Python is dynamically typed, you can use type annotations to hint the type of a variable. This is optional and mainly used for documentation and type checking.

```
# Type annotations
my_variable: int = 42
```

7. Constants (Convention):

While Python doesn't have constants, it's a convention to use uppercase names for variables that should be treated as constants.

```
# Constants (by convention)
PI = 3.14
```

8. Global and Local Variables:

Understanding the scope of variables is important. Variables declared inside a function are considered local unless explicitly stated otherwise.

```
# Global variable
global_var = 100
def my_function():
    # Local variable
    local_var = 50
    print(global_var)  # Accessing global variable
    print(local_var)   # Accessing local variable
my_function()
```

In Python, variables are created when you first assign a value to them, and their type is determined dynamically. This flexibility contributes to the simplicity and readability of Python code.

Examples of integers, floats, strings, and booleans:-

Certainly! Here are examples of integers, floats, strings, and booleans in Python:

1. Integers:

```
# Example of integers
age = 25
```

e-Book

Inside this book

"Play with Python" is your comprehensive companion for diving into the world of Python programming. Through a blend of hands-on examples, stimulating projects, and thought-provoking exercises, you'll embark on a journey of skill acquisition and mastery. Whether you're a beginner eager to grasp the fundamentals or a seasoned coder looking to expand your repertoire, this book caters to all levels of expertise. Explore the versatility of Python as you delve into web development, data analysis, automation, artificial intelligence, and beyond. Each chapter builds upon the last, guiding you through a structured learning experience designed to reinforce concepts and foster proficiency. Along the way, you'll discover the joy of problem-solving and the satisfaction of creating practical solutions using Python. With its accessible approach and practical focus, "Play with Python" ensures that learning to code is not only educational but also enjoyable. So, grab your keyboard, ignite your curiosity, and embark on a rewarding journey of exploration and discovery with Python.

Mr. Pawan Kumar

Dr. Anupa Sinha

Mrs. Akanksha Mishra

Mrs. Sushree Sasmita Dash

ISBN 978-93-340-0763-3

9 789334 007633

Innovation and Integrative Research Center Journal

ISSN: 2584-1491

Address:- Kota, Raipur, Chhattisgarh, India.

492001, Contact No.- +91-8871386512

Email: connect@samagracs.com