MASTER OF COMPUTER APPLICATION

SEMESTER 1

PYTHON PROGRAMMING

Unit 1

Introduction to Python

Table of Contents

1. INTRODUCTION

Welcome to Unit 1, where we'll embark on an enlightening journey through the Python programming language, a tool that has become indispensable in today's competitive tech landscape. Python's allure lies in its dual role as a high-level programming language and a rapid application development tool, making it an ideal choice for designing and supporting large applications. This unit will unravel Python's simplicity and dynamism, showcasing why it's the preferred language for web applications and object-oriented programming. We'll delve into Python's history, tracing back to its inception in 1991 by Guido van Rossum, and explore how it evolved to simplify programming tasks by reducing time spent on redundant activities. By understanding Python's core principles and applications, from web development to data science, you'll appreciate its role in powering platforms like YouTube and Dropbox and why top-notch companies rely on it for cutting-edge applications.

Approaching this unit requires a blend of curiosity and practical engagement. Start by absorbing the historical context and foundational concepts of Python to grasp its versatility and ease of use. As you progress, focus on the real-life applications of Python, connecting theoretical knowledge with practical utility. Experiment with Python by writing simple scripts, exploring its vast libraries, and engaging with the community-developed frameworks like Django and Flask. This hands-on approach, complemented by exploring Python's rich ecosystem, will not only solidify your understanding of the language but also inspire you to leverage Python's capabilities in your projects. Embrace this journey through Python, and you'll soon find yourself adept at navigating the world of programming with confidence and creativity.

1.1. Learning Objectives

At the end of this topic, you will be able to.

- *Describe the evolution and key features of Python to understand its significance in modern programming.*
- *Explain how Python's applications across various domains like web development, data science, and machine learning make it a versatile tool for developers.*
- *Apply Python's basic syntax and libraries to create simple scripts, demonstrating an understanding of its ease of use and flexibility.*
- *Analyse the differences between Python 2 and Python 3 to make informed decisions when choosing a version for project development.*

2. ABOUT PYTHON LANGUAGE

The competitive times call not just for a high-level programming language, but also for a rapid application development (RAD) tool. Such tools should be able to design and support large applications seamlessly while providing features like fast response and a simple approach. All such needs are fulfilled by Python.

Python is one of the few languages that can be used as both, an interpreted and scripting language. It is a free and open-source language that is easy to learn and approach. Especially suited for Web applications, Python makes object-oriented programming simple and dynamic, enabling one to create large applications that are durable and adaptable.

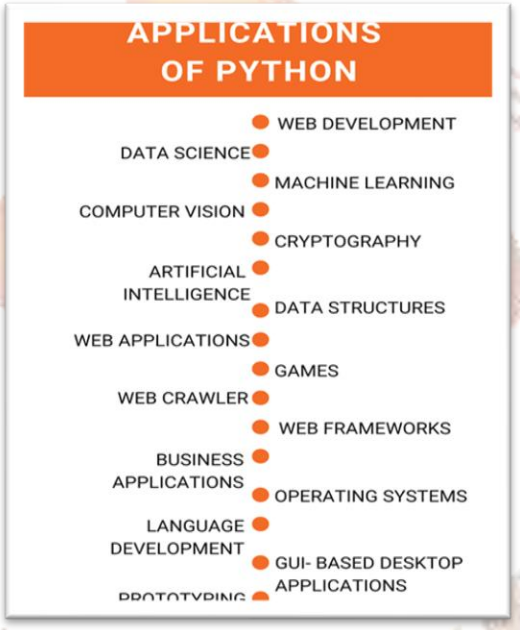
Python was developed in 1991, with the aim that it could help reduce the time spent on monotonous and redundant tasks in programming. It is derived from the original scripting language ABC. At that time, ABC was used solely to teach programming by a few people.

Developed by Guido van Rossum, Python was originally created to perform and handle regular administrative tasks. It was employed at CWI in 1989, where Guido van Rossum was located at that time. It was released for the use of public in February 1991, after it was integrated into the Amoeba project at CWI.

Since then, the development of the language has been carried out at CNRI in Reston, Virginia, United States. By 2000, the developers of Python formed Python labs under the BeOpen network. Python labs has been maintained and organised by Guido van Rossum, among other primary developers of Python. Later in the year, Python labs moved under Digital Creations. It has since been working to develop and enrich the language with more and better features. All the features, properties, etc., that are added to the language are managed by Python Software Foundation, a non-profit organisation.

3. NEED AND APPLICATIONS OF PYTHON LANGUAGE

Python is the backbone of various web-based applications such as YouTube, Dropbox, etc. Its functionality and features such as supporting cross-platform operating systems make it a great tool for various applications. Here are the real-life applications where Python is needed and highly used by top-notch companies.



Source: Pinterest

Fig 1: Applications of Python

3.1. Web Development

Python supports various frameworks. This increases the security, scalability, and speed at which a web-based application or website is developed and coded. There are multitudes of frameworks that support the integration of HTTPS, FTP, SSL, and much more. Frameworks like Flask, Django, and more can help the processing of JSON, E-Mail, XML, and other such protocols.

3.2. Machine Learning and Artificial Intelligence

The future is in machine learning and artificial intelligence. It is imperative to learn and develop languages that can enhance these technologies and Python supports the experience.

For machine learning, the computer should be able to come up with an algorithm based on its experience and learn by itself. This can be done by libraries such as NumPy, Pandas, and more which are already present in Python.

3.3. Data Science

Understanding data and visualising it can help avoid risk and increase profits. The computer should be able to understand which data is relevant for a task, analyse it, and provide the results accordingly. Through libraries such as Pandas and NumPy, Seaborn, and Matplotlib, you can visualise and analyse the data. Python is an unmatched tool for data scientists.

3.4. Game Development

Python also includes libraries that support 3D game engines such as PySoy, PyGame, etc. These libraries ease and enhance the process of building a game. Python also ensures that the game processes faster and without any bugs and errors.

3.5. Business Applications

Business applications should be scalable to support the business as it grows. They should also be readable and extensible. Thus, languages like Python provide all such features to the developers. Various platforms such as Tryton can create applications that are coded in Python.

3.6. Desktop GUI

Using the Tkinter library available in Python, you can create desktop applications as well. From simple applications such as post-it notes and calculators to high-end applications, toolkits such as wxWidgets, PYQT, and others provided in Python can help you create and develop various such applications.

3.7. Web Scraping Applications

There are various applications such as job listings, research and development, comparison of prices, etc. when one needs to assess and analyse data from various websites and huge

databases. Python's libraries such as BeautifulSoup can scrape and analyse such data without compromising speed and efficiency.

Along with these, Python is used to develop various other applications such as audio and video supporting apps, CAD applications, embedded applications, and much more. Its flexibility and scalability make it suitable for high-level computations and complex codes.

4. EVALUATION OF PYTHON LANGUAGE

When comparing Python with other languages, the one benefit that makes Python unique is that it works as a bridge between interpreted and scripted languages. It holds the benefits of both types of languages, making it a much more accessible option than its other counterparts.

Most scripted languages are slower than interpreted languages. However, Python provides the user with the option to write scripts using Python without compromising on the speed of the compiler. Python is usually compared to C and C++ because of its similar syntax. It is mainly used to test applications written in C and C++. Python works better than these two high-level languages as it provides various benefits over them such as memory allocation and management, avoiding reference errors. The Python interpreter is responsible for memory management. Also, with the rich library available in Python, you can write much shorter and compact programs.

Another such language that can be compared to Python is Perl. Perl is another popular language that is used for data extraction and text manipulation. It is mostly used for system administration. However, Perl follows a difficult and much more complex syntax as compared to Python. Perl can be used to create powerful CGI (Common Gateway Interface) scripts, but the code is difficult to comprehend and people working on a large project often find it hard to understand each other's code.

Tcl is also one of the programming languages that is often compared to Python. Tcl is as powerful as Python and is an easy-to-use scripting language. However, the data types for variables in Python are much richer than those available at Tcl. Tcl and Python have a similar toolkit for developing GUI applications, called Tk. However, though people tend to favour Python while developing such applications for its richer library.

Python is an object-oriented language that works similarly to Java. However, Java requires larger codes that take longer to write, understand, and compile. It also does not offer features such as a rapid development environment and dynamic typing. But Python is slower than Java and is much less portable. To integrate the features of Python and classes available in Java, JPython was developed. JPython provides various salient features such as providing a true object-oriented programming environment and a scripting environment for Java.

5. PYTHON 2 VS. PYTHON 3

Python 2 was first developed and released in 2000. It has been one of the most successful versions of Python, the most popular one being Python 2.7. It was released in 2010 and is still in use by various companies.

Source: DEV Community

Fig 2: Python 2 vs Python 3

Python 3 was created in 2008. At that time, most programmers and companies were already using and were satisfied with the previous version of the language. However, the tools and features added to Python 3 soon spread amongst the community, leading to a debate among the programmers regarding which is the better version.

With the constant development and betterment of Python 3 and the strength of version Python 3.5, it was clear that the newer version of Python is the clear winner of the debate and should be the choice of new programmers who are just delving into the world of Python.

Now, most programmers agree that Python 2 dictates the legacy and the strength of the programming language whereas for the newer and faster web-based applications suited to the speed of the current generation, Python 3 and higher versions are much more suitable. Here is a comparison between both versions to aid your understanding.

5.1. Incompatible Libraries

The libraries that were developed for Python 2 are incompatible with Python 3. Also, seeing that Python 3 is the future, the libraries developed are strictly compatible with this version only.

5.2. Unicode Support

In Python 2, strings were stored in ASCII by default. If you want to store a string in Unicode, then you will have to specify a "u". However, in Python 3, all the strings are stored in Unicode by default. Unicode is much more versatile as compared to ASCII. It can also save various symbols, numerals, emojis, Roman numerals, and much more.

5.3. Better Integer Division

In Python 2, if you divide two integers that will result in a decimal answer, then the compiler will round off the answer to the nearest integer. Hence, to get an answer in decimal form, you will have to specify decimals in the subtraction. For example, use 2.0 instead of 2.

In Python 3, the compiler is expected to return the result in decimal if the answer is decimal. You do not have to add a decimal to the question.

Various companies such as Instagram and Facebook are moving from Python 2 to Python 3. This is because the newer function has a faster runtime. Also, there will be no developments in Python 2 after the version Python 2.7, making it a thing of the past.

6. FEATURES OF PYTHON PROGRAMMING MEASUREMENT OF SOCIAL

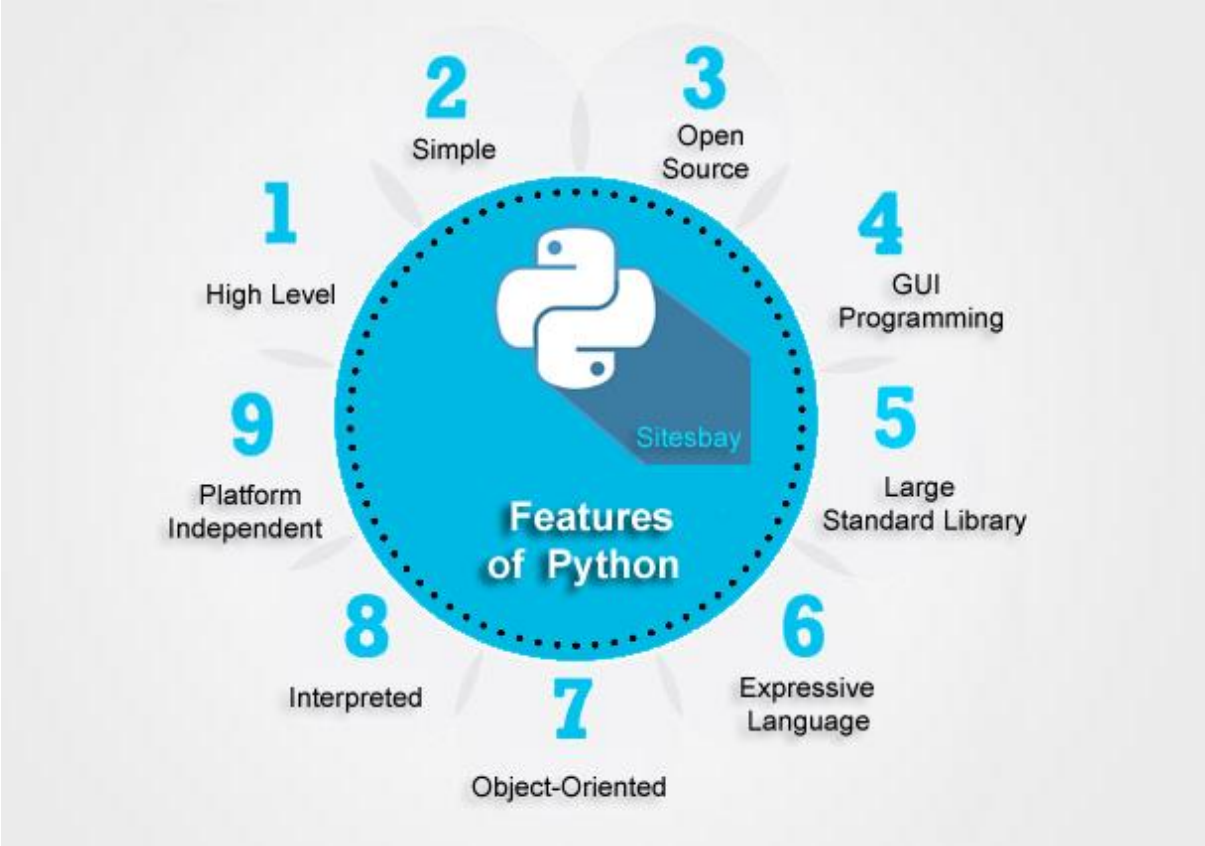
The best feature of Python can be its ease. Programmers can pick Python as their first language to learn. You can easily develop large applications using it. It can also act as a bridge to connect the programmers to other, more advanced programming languages. Here are some of the prominent features of Python that make it a flexible and strong option for most developers.

6.1. Easy to Understand and Use

You can understand code written in Python even if you have no background in programming. It has an easy syntax that uses the common language without the presence of complex keywords. The lack of too many keywords and simple semantics used in Python ensures that one can learn it with relative ease. It is a simplified version of primary programming languages such as C, Algol, Pascal, etc., and follows the same syntax.

Though an object-oriented language, it is not necessary to use the concept in Python. One can run various applications and operations without it, a feature that makes it easier to approach than C++.

Python uses very few symbols within the code, to make coding simple and straightforward. It does not need a symbol to show the end of a command, nor does it need curly braces ({}) to indicate a block of statements. One can use indentation to group statements as a single code block, ensuring fewer bugs and easier comprehension of the code.



Source: Sitesbay

Fig 3: Features of Python Language

6.2. High-Level Language

Python is an intuitive language. While using C and other such languages, one must declare the datatype of the variable before storing any data in it. Also, other features such as system calls, strings, or varying lengths, etc., need a chunk of code to be written in C. Python reduces the time and energy spent on understanding and writing the code by providing built-in modules to its users. Lists, dictionaries, and other high-level data types that can be used in Python are already built into Python. Thus, using them requires less time and understanding of the language. Also, Python assumes the datatype of a variable once you assign a value to it, decreasing the length of the code.

6.3. Scalable

Scripting languages such as Unix work seamlessly and are easy to use while performing easy and straightforward tasks. However, once you start adding more lines to the code or features in the script, it becomes complicated and slow. These days, even smaller applications need large code and scripts to include all the required features. Hence, these scripting languages cannot keep up with today's need for fast and scalable programming languages. Through Python, one can merge one code with another or insert new features in an existing code without worrying about the speed of the code. Python ensures more functionality by allowing the user to change and modify the code by adding plug-ins and advancing the architecture of an existing script.

6.4. Object-Oriented Language

An Object-oriented language means that each component in a code written in Python is treated as an object. Though not necessary to use the concept of OOP (object-oriented programming) in Python, it is nonetheless a strong tool. You can create smaller programs without indulging or understanding the concept of OOP, making it easier to learn and understand the language. For larger programs, you can use object-oriented class hierarchies with every attribute in the class possessing a name. Python can accept any number of arguments.

6.5. Interpreted Language

In conventional interpreted languages, the program is not interpreted in binary language. Thus, the process takes longer and is slower than other languages that support the compiler. Though, Python is an interpreted language where code is already byte-compiled, that is, available in binary language when it is added to the interpreter. The feature enables Python to act as an intermediary between compiled and interpreted language, as it has the features of both.

Python is an interpreted language as the interpreter can print the result when in command-line mode. Signs such as ">>>" work as a prompt, indicating that the interpreter is ready to use.

Another way a program can be interpreted is through executing a file. Such a file is called a script and is saved with an extension of .py. To execute the file, the name of the script must be written for the interpreter to understand.

6.6. Flexible

A large code can be hard to understand and debug. Python allows fragmenting the code into smaller, more manageable parts. These parts, or modules, can still interact with each other and connect with the other in-built modules. Python allows flexibility to the user by letting them interact with the external environment while using the module.

6.7. Library

Python comes with a rich library of modules that can be integrated into the program. The library is updated and maintained by Python Standard Library. These modules can ease the process of coding and shorten the length of the code as well. The library can be used to write codes that perform the tasks of HTTP, FTP, POP, etc. You can write applications for various functions with the help of the built-in modules of Python that are used to develop graphic user interfaces, download a webpage, and various others.

6.8. Memory Management

Memory management is an essential part of programming when it comes to languages like C, C++, etc. Without proper memory management such as deleting the assigned memory once the task of the variable is completed, the programmer may face issues like memory leakage. This makes the program unnecessarily long. In Python, the task of memory management is handled by the interpreter. The interpreter is bound to make fewer or no errors and decrease the development time of the program as well.

6.9. Web Scripting Support

Python is a dynamic language. It can not only handle complex programs with ease but also supports several environments. Web applications based on the internet and intranet need such features in a programming language for them to work smoothly. Through the rich

library system and flexibility of Python, you can write several complex programs and codes that support advanced features such as HTML, SGML parsing, XML, CGI scripts, etc.

6.10. Database

Through the built-in modules available in the library of Python, you can handle several flat file databases. It has the capability to provide interfaces for major databases. One of the most essential and multi-featured databases of Python is Python API. It eases the process of writing codes for applications through which you can communicate with different databases.

6.11. Object Distribution

Through Python, you can use objects that are distributed over various platforms. The code can communicate and interact with objects that are coded or available in other languages. One such example is passing data to COM components.

6.12. Embeddable

Python interpreter can be extended with low-level modules. These extensions let the programmer customise the program according to the needs and interface of the application. You can connect Python to external libraries and modules to implement new data types in the code. Extension modules on Python can be programmed in other languages such as C and C++ for CPython and in Java for JPython.

6.13. GUI Programming

Python holds a graphic user interface library Tkinter that allows the user to create GUI applications for different systems and libraries such as Windows MFC, X Window system of Unix, Macintosh, and more. Tkinter has the object-oriented interface of TK GUI API.

6.14. Portable

Python is written as ANSI C. Hence, it is not tied to one operating software and can be used over many platforms without losing any of its features or credibility. One can write, execute, test, and upload a program written in Python in various environments including Macintosh, Linux, and Windows. However, the application will run as per the commands specified in it.

If the commands are specified for one environment, for example, Linux, then the program will not be able to implement properly when interpreted in other environments.

6.15. Freeware

Python is an open-source language and thus can be redistributed freely. Anyone can use Python's source in any way they want without harming or putting it at risk. Also, the users are dissuaded from trying to take over the copyright of the source code. Programmers and users around the world can freely use Python to create codes using modules from the library in byte-compiled form.

6.16. Exception Handling

Through exception handling, Python can generate a stack trace of errors. This is when due to an error or bug in the program, Python exits. With the help of the trace, you can detect errors during run time without the need for statements that can check errors. The programmer does not have to spend a lot of time debugging the code as the Exception Handler can detect the problem, diffuse it, and perform a maintenance check.

6.17. Popular Frameworks of Libraries

Python is mainly used to write various web-based applications. To make the process of coding easier, various modules are created and stored in the library of Python. The collection of these modules is called a framework. Frameworks allow one to integrate codes for low-level details and help the programmer to save time.

By using various frameworks, one can automate the implementation of common commands, thus giving the user the time to spend on web-based application logic and algorithms.

Here are some of the most popularly used frameworks in Python.

6.18. Django

It is an open-source and free framework. Based on the DRY principle, or Don't Repeat Yourself, Django uses ORM mappers. Object-relational mapper is used to manipulate the data from a database. It uses an object-oriented paradigm. Through Django, one can get various

essential features in their code such as ORM, template engine, URL routing, authentication, and database schema migrations.

6.19. Web2Py

It is another full-stack framework for Python that is scalable and open-source. It can be installed without any special prerequisites. Web2Py can be implemented and run in various environments and different platforms. It protects the code against cross-site scripting, SQL injection, etc. It comes with its own code editor, and debugger, and has a one-click deployment as well. It does not support Python 3 yet.

6.20. Flask

This lightweight micro-framework is customisable and adapts to the needs of the user. It provides multitudes of features such as a quick and seamless debugger, a built-in development server, and secure cookie support. It also supports HTTP request handling, can plug in any ORM, supports Jinja2 templating, and is Unicode-based.

6.21. Bottle

The bottle provides default features such as templating, routing, a built-in development server, and an abstraction layer over the WSGI standard. It is a micro-framework whose server can support all other WSGI-capable HTTP servers. It can also access data to create file uploads, cookies, headers, etc.

6.22. CherryPy

CherryPy is especially targeted to ease the process of creating web-based applications. It is an open-source framework that makes coding for web-based applications like object-oriented programming. It packs in features such as caching, authentication, encoding, plug-in system flexibility, and the ability to run on different platforms and in different environments. It can run multiple HTTP servers simultaneously and offers built-in support for profiling, testing, and coverage.

Different frameworks are suited for various applications. All offer different functionalities that can be chosen as per the purpose of the program. You will have to assess the need for the program and choose the framework that provides the required features.

7. SYSTEM REQUIREMENTS

You need to ensure that your system meets all the requirements before you proceed with the installation of Anaconda. Depending on whether you are using the free Anaconda distribution or are an Anaconda Enterprise 4 user, the system requirements will be different. Let us look at the system requirements for both one by one-

Source: mrmint.fr

Fig 1: Anaconda

7.1. System Requirements for Anaconda Enterprise 4

7.2. Hardware Requirements

Without satisfying the hardware requirements, it will not be possible to move ahead with the installation. The requirements are-

- The RAM should be 32 GB or 16 GB of 1600 MHz DDR3 RAM.
- A CPU having the following features: 2 x 64-bit 2.8 GHz 8.00 GT/s CPUs
- Minimum storage of 300 GB. In the case of air-gapped deployments, the minimum storage size should be 600 GB.
- The user should have internet access to download all the files from the website Anaconda.org. Alternatively, the user's laptop or PC should have a USB drive containing all the required files. In the case of air-gapped installations, the user should also have alternate instructions.

7.3. Software Requirements

The minimum software requirements are essential to ensure the smooth functioning of the application. They are-

- The system should have MongoDB 2.6. It will be provided with the installation files.
- The user should have the Anaconda Repository license file.
- RHEL/CentOS 6.5 to 7.4, Ubuntu 12.04+. Additionally, Ubuntu users might need to install cURL.
- The user should have a Linux system account. The following are required- Mongod (RHEL) or MongoDB (Ubuntu)
- The system should have access to the Anaconda server.
- Cron entry is required to start the repo on reboot.

7.4. Security Requirements

The security requirements ensure that the data is protected, and no data breach takes place. The security requirements are-

- An Open HTTP(S) port
- Edit privileges for SELinux policy (SELinux does not have to be disabled in the case of Anaconda Repository operation)
- Ability to make Iptables modifications. It is an optional requirement and depends on the user.
- The user can choose to have an SSL certificate. It is not mandatory to have it.
- Privileged access OR Sudo capabilities

7.5. Network Requirements

TCP ports will be required. Most of the below requirements are optional, and it is up to the user if he considers them as must-haves.

- Inbound HTTP: TCP 8080, 8443 (Anaconda repository)
- Optional Inbound SSH: TCP 22 (SSH)
- Optional Outbound HTTPS: TCP 443

  - repo.anaconda.com
  - anaconda.org
  - conda.anaconda.org
  - binstar-cio-packages-prod.s3.amazonaws.com
  - 820451f3d8380952ce65-4cc6343b423784e82fd202bb87cf87cf.ssl.cf1.rackcdn.com
- An outbound Simple Mail Transfer Protocol: Transmission Control Protocol 25 (if not using AD/LDAP) email notifications. It is not compulsory to incorporate it.
- An outbound LDAP(s): TCP 389/636 for authentication integration. It is not a mandatory requirement.

7.6. Other Requirements

- A license file provided to you by Anaconda at the time of purchasing the product.
- The installation tokens for Binstar and anaconda-server channels were provided by Anaconda while purchasing the product. It is not applicable for air-gapped installs.
- It is an optional requirement: Your Anaconda.org account credentials. It is not applicable for air-gapped installs.

7.7. System Requirements for Anaconda Individual Edition

It is mandatory to review the system requirements before installing the Anaconda individual edition. The requirements are-

- A minimum of 5 GB disk space is required to download and install Anaconda.
- License: It can be the one for free use and redistribution under the terms of the ../Eula.
- Operating system: Windows 8 or the latest one, 64-bit macOS 10.13+, or Linux, including Ubuntu, RedHat, CentOS 6+, and others.
- If your operating system is older than what is currently supported, you can find older versions of the Anaconda installers in the archived section.
- The system architecture should have the following features: Windows- 64-bit x86, 32-bit x86; MacOS- 64-bit x86; Linux- 64-bit x86, 64-bit Power8/Power9.

8. INTERACTIVE MODE AND SCRIPT MODE

Python offers two primary modes of execution: interactive mode, ideal for exploratory programming and immediate feedback, and script mode, suited for developing complex applications and automating tasks. Both modes cater to different aspects of programming, making Python a versatile tool for both learning and professional software development.

8.1. Interactive Mode

Interactive mode in Python is an incredibly powerful feature that fosters an exploratory approach to programming, allowing for immediate feedback and facilitating a deeper understanding of Python's functionalities. This mode, often accessed through Python's shell (REPL - Read, Evaluate, Print, Loop), is instrumental for rapid prototyping, debugging, and learning, as it executes Python commands as they are entered, line by line. Unlike script mode, where programs are written in a file and then executed, interactive mode is more dynamic, enabling users to experiment with code snippets, test functions, and observe results instantaneously. This immediacy is invaluable for beginners to grasp programming concepts and for seasoned developers to test complex code snippets without the overhead of creating a full-fledged program.

Interactive mode also serves as a powerful calculator, allowing users to perform mathematical operations and experiment with Python's vast standard library, third-party modules, and various frameworks. It's particularly useful for data analysis and scientific computing, where users can import libraries like NumPy and Pandas and interact with data in real-time, visualizing results, tweaking parameters, and refining algorithms on the fly. The mode supports introspection, enabling users to explore objects, query types, and access documentation directly within the session, thereby enhancing the learning curve and productivity.

Furthermore, interactive Python environments like IPython and Jupyter notebooks extend the capabilities of the standard interactive mode with additional features like advanced auto-completion, session history, and inline plotting. These tools have transformed the way Python is used for data science, making it more accessible and interactive. They allow for a narrative approach to coding, where code, output, and documentation can be intertwined in

a single document, fostering a more intuitive and engaging learning and development process.

Python's interactive mode is not just a feature but a gateway to a more interactive, exploratory, and efficient programming experience, aligning with the dynamic and versatile nature of Python itself. It encapsulates the spirit of discovery inherent in the programming journey, making Python an inviting and powerful language for developers and data scientists at all levels of expertise.

8.2. Script Mode

Script mode in Python, in contrast to its interactive counterpart, is where the true power and efficiency of programming with Python come to the forefront. This mode involves writing Python code in a file (typically with a .py extension) and then executing the entire script in one go, either through a command-line interface or an integrated development environment (IDE). Script mode is the backbone of software development in Python, enabling programmers to build complex, reusable, and maintainable code bases. It's particularly suited for developing applications, automating tasks, and scripting complex computational tasks that go beyond the scope of one-off commands or immediate feedback loops offered by the interactive mode.

In script mode, programmers can leverage the full spectrum of Python's features, from basic syntax to advanced concepts like object-oriented programming, exception handling, and module packaging. This mode encourages structured programming practices, including the use of functions, classes, and modules, which are essential for breaking down complex problems into manageable units, promoting code reuse, and enhancing readability. Scripts can be as simple as a few lines of code to automate mundane tasks or as complex as thousands of lines forming the backbone of robust applications.

Moreover, script mode facilitates collaboration among developers through version control systems like Git, enabling teams to work on different parts of a project simultaneously, track changes, and merge code seamlessly. It also allows for thorough testing and debugging, essential for ensuring code reliability and performance. Tools such as unit tests can be

integrated into the development process, allowing for automated testing of code functionalities.

For tasks that require running algorithms repeatedly, perhaps with varying parameters or data sets, script mode is invaluable. It allows for batch processing, where scripts execute tasks without manual intervention, saving time and reducing the potential for human error. This mode is fundamental for deploying Python applications in production environments, where scripts are often scheduled to run at specific times or triggered by specific events, ensuring that applications perform their functions reliably and autonomously.

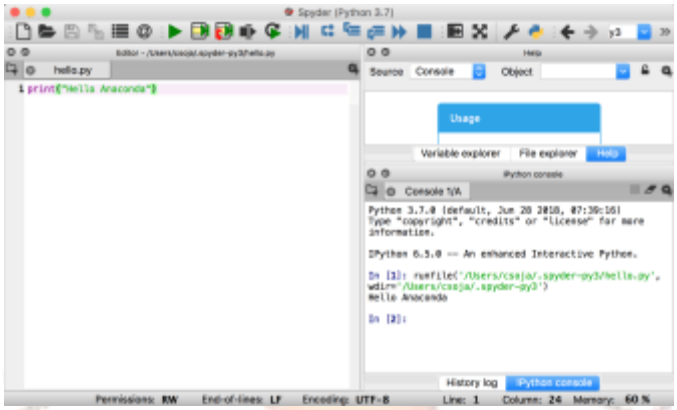
We can summarise that script mode is where Python's versatility as a programming language truly shines, enabling developers to build everything from simple scripts to complex, scalable applications. It embodies the principles of software engineering, making Python an indispensable tool for developers across diverse domains.

9. INSTALLATION OF ANACONDA

Once you are done verifying the system requirements, the next step is to start with the installation process. If you do not want several packages to be installed with Anaconda, you can choose to install Miniconda. It is a smaller version of Anaconda that includes Python, Conda, and its dependencies.

On macOS, Windows, and Linux, it will be the most convenient to install Anaconda for the local user. Thus, administrator permissions will not be required. The following are the steps to install Anaconda on your system-

1) Go to the https://www.anaconda.com/products/individual link to download Anaconda for your respective OS (macOS, Windows, or Linux). You can download the installer either for Python 3.7 or Python 2.7 (whichever is the latest). Also, you can download it for a 32-bit machine or a 64-bit machine as per your machine's specifications.
2) The Anaconda setup would have downloaded in .exe format. Click on the file to open. Click on the 'next' button.
3) You should reach the license agreement page. Read the agreement click on 'I Agree' and move to the next page.
4) You can choose to install Anaconda only for yourself or for all users. You will need administrative privileges to install it for everyone.
5) Choose the folder where you want to install it. You will be able to see the available space on the system and how much is required by the application.
6) Now, you reach the screen of advanced options. Firstly, you need to add Anaconda to the PATH environment variable of the system. Now, register it as the primary system Python 3.7. If you add Anaconda to the PATH, it will be found before any other installation. Click on the "Install" button.



Source: Francesco Lelli

Fig 2: Anaconda Editor

7) It will take some to extract the files and unpack the packages on the machine. The installation will be complete in some time. Click on the 'next' button
8) The next screen will inform you about PyCharm. Click on the 'Next button. The installation is complete. Click on the 'Finish' button.
9) Now, you will be able to see Anaconda in the Start menu. You can open a Jupyter notebook and create a new notebook.

10. ANACONDA NAVIGATOR

The Anaconda Navigator is a Graphical User Interface (GUI) that allows you to conveniently manage environments, Conda packages, and various channels without the need to use command-line commands. The navigator is included in the Anaconda distribution and allows you to launch various applications. Navigator can be used to search for packages on the website Anaconda.org, or in a local Anaconda Repository. It is available for macOS, Windows, and Linux.

10.1. What is the use of the Anaconda navigator?

To run, several scientific packages are dependent on versions of other packages. Data scientists often use several versions of many packages and use distinct environments to separate these different versions. The command-line program Conda is both an environment manager and a package manager. This feature helps data scientists ensure that each version of every package has all the dependencies it needs and works as per expectations.

Navigator is a convenient, GUI way to work with environments and packages without having to type Conda commands in the terminal window. You can use the navigator to find the packages you are looking for, install them in the environment, run the packages, and update them.

10.2. What does the application Navigator allow us to access?

By default, the following applications are available in the Anaconda Navigator:

- JupyterLab
- Jupyter Notebook
- VSCode
- RStudio
- Spyder
- Glueviz
- Orange 3 App
- PyCharm

- Anaconda Prompt (Windows only)
- Anaconda PowerShell (Windows only)

User having an advanced level of experience in Conda can build their applications as well. The simplest way to run a piece of code in Navigator is using Spyder. Navigate to the navigator home tab and click on Spyder. You can write and execute your code.

10.3. How to start the Anaconda Navigator?

The process to start the Navigator depends on the Operating System. Follow the below steps to start the navigator in different Operating systems:

1. Windows: Click on the Anaconda Navigator Desktop app from the Start Menu. Another way could be to search for Anaconda Prompt from the Start Menu and then type in the command anaconda-navigator.
2. Linux: Type in "anaconda-navigator" after opening the terminal window. The Navigator will open.
3. MacOS: Click on the Anaconda Navigator icon from the launchpad. Alternatively, you can open the terminal from the launchpad and type in anaconda-navigator.

11. DIFFERENT MODULES OF ANACONDA

Modules refer to a file containing Python definitions and statements. Any file containing a Python code, for example, Abc.py, is known as a module, and its module name would be ABC.

Any package in Anaconda can be installed using the command "Conda install Package name". All packages are in the package repository. Apart from the packages, the major components of Anaconda are listed below-

11. 1. Jupyterlab

JupyterLab is a web-based UI (User interface) developed for Project Jupyter. JupyterLab allows users to work with activities such as Jupyter notebooks, documents, terminals, text editors, and several other components in an integrated and flexible manner.

One of the biggest advantages of JupyterLab is the option to arrange several documents side by side in the working area using splitters and tabs. Various activities and documents integrate seamlessly and thus enable the creation of new workflows that act as a push for interactive computing. A few examples of the same area-

- The Code Consoles provide temporary scratchpads for executing and running the code interactively. It behaves as a major support for rich output. A code console can be connected to a notebook kernel which will act as a computation log from the notebook.
- Documents that are kernel-backed allow code in any format of a text file (Markdown, R, Latex, or Markdown) to run in a Jupyter kernel interactively. Outputs of notebook cells can be mirrored into a tab of their own. These tabs can be side by side with the notebooks. Simple dashboards are enabled with interactive controls along with a kernel.
- Multiple views of documents are allowed where viewers enable live editing of their documents which will be reflected in the documents of other viewers.
- Another advantage of JupyterLab is that it allows handling and viewing data formats in various file formats like JSON, PDF, and CSV. You will also get the option to use key maps and customizable shortcuts on the keyboards.

11.2. Jupyter Notebook

The Jupyter Notebook is popular for extending the console-based approach to interactive computing in a new direction. It provides a web-based application that can capture the end-to-end computation process. The process involves documenting, developing, and executing the code followed by communicating the results to the users. The Jupyter Notebook comprises two components:

1. A web application: It is a browser-based tool made to author interactive documents that are a combination of computations, mathematics, explanatory text, and their media output.
2. Notebook documents: It is a representation of the content seen in the web application. It also includes the input and output results of the computation, images, representation of objects, mathematics, and explanatory text.

Notebook documents contain both the input and output of an interactive session in addition to the text that accompanies the code and is not meant for execution. Thus, Jupyter Notebooks can be considered as a computational record of the sessions. It also includes mathematics, explanatory text, and executable code. These documents are JSON files and can be easily shared with colleagues. As we are using Jupyter in a browser, our computer will be acting as the server. Hence, Jupyter does not need to send our data elsewhere.

11.3. Spyder

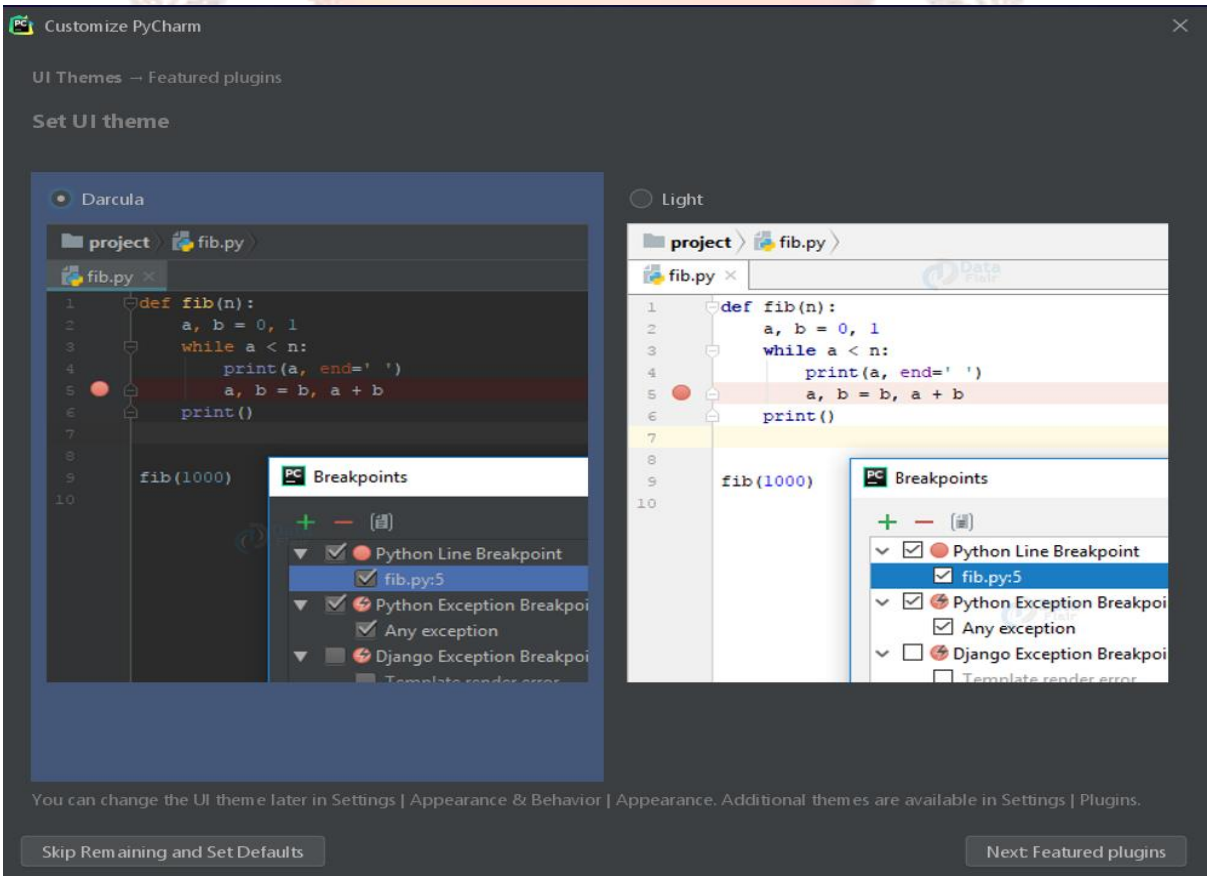
It is an open-source IDE (Integrated Development Environment) that has cross-platform integration to make scientific programming in Python easier. Several packages are integrated with the scientific Python stack. A few of them are SymPy, pandas, Cython, NumPy, and Matplotlib. Spyder is extensible with third-party plugins and supports interactive tools for data inspection. It also takes care to embed Python-specific introspection instruments and code quality assurance instruments. The primary features of Spyder are-

- It provides support for several IPython consoles.
- An editor with code completion, syntax introspection, and highlighting.
- A history log that takes note of every command that the user enters.

- A static code analysis.
- Spyder allows users to work on multiple development efforts concurrently.
- A help pane that can retrieve and render documentation on classes, functions, and methods.

11.4. Pycharm

PyCharm is another popular IDE used for creating scripts in Python. It provides various essential tools that enhance the experience of developers while using Python. They provide an integrated environment to enhance the productivity of app development in Python. It is available in editions. The free and open-sourced one is called the Community version.



Source: Data Flair

Fig 3: A simple program on PyCharm

Another free version is also available called Edu. It is used to learn programming languages for other technologies related to educational tools.

The paid version of PyCharm is called Professional. It is widely used by developers for code assistance, visual debugging, remote configurations, assistance, etc. The salient features of PyCharm are explained below.

- Code completion: Through PyCharm you can experience intuitive code completion for built-in and external packages alike.
- Code coverage: Files with extension .py can be run outside of the PyCharm editor as well.
- Package Management: The packages installed are displayed with appropriate visual representation.
- Refactoring: Through refactoring, one can name one or more files at the same time. PyCharm makes this process much smoother.
- Local History: PyCharm keeps track of the changes made in a file locally.

11.5. Vs Code

VS Code or Visual Studio Code supports code and development in various programming languages through its extension model. You can use the Python extension to use the VS Code for developing programs and scripts in Python. Here is a brief list of features provided by the VS Code when used for Python.

- It supports all versions of Python, including the classic Python 2.7 and the newer versions of Python 3.4 and above.
- It offers code completion with its unique IntelliSense.
- It enables the automatic use of virtual environments and conda.
- You can use Jupyter environments and Jupyter Notebooks for code editing.
- Visual Studio Code also provides themes through which you can customize the UI as per your choice.
- Varied language packs ensure that the experience of working on the VS Code is inclusive and localised.

- Code snippets
- Debugging Support

11.6 . Glueviz

Glue has become an integral part of the complete Python programming experience with its indispensable features and the ability to explore relationships between related datasets. One can explore the environment of object-oriented programming with the help of Glue. It is used to analyse astronomical data such as the formation of stars through clouds and comprehend medical data. prominent features of Glue are listed below.

- Linked Visualizations: With the help of Glue, you can visualise data in the form of histograms, 2D, and 3D images. scatter plots, etc. It also supports the brushing and linking paradigm. Thus, if data is selected in a graph, then it propagates to other sets of data.
- Scripting Ability: You can integrate Python codes into Glue seamlessly. The Glue itself is written in Python. The integration can help in cleaning, inputting, and analysis of the data.
- Data Linking: Glue has immense potential when it comes to linking two or more sets of different data. It uses logical links that relate to these data and with the help of them, visualizes it. The links can be specified by the programmer and are flexible.

11.7. Orange 3 App

Being able to analyse and comprehend huge programs and data is the feature that makes Python so much more desirable than other programming languages. Orange 3 supports this feature of Python by providing a clean, free, and open-source platform that is powerful holds the potential to analyse large data and provides comprehensive visualizations for the chosen data. Orange 3 also supports basic machine learning skills and analyses.

The features of Orange 3 have been a boon for various fields of research where scientists need to analyse the data obtained from a multitude of experiments. Here are some of its features that make it an easily approachable option.

Interactive Visualization: You can scatter plots, box plots, make decision trees, hierarchical clustering, linear projections, heat maps, and more.

Visual Programming: You do not have to dive deep into the coding part to make it work. The graphic user interface includes widgets that make analysing datasets much easier.

11.8. Running A Simple Code Python Using Jupyter and Spyder

Now that the setup of the Jupyter Notebook is complete, let us try to run a simple program on it. Here are the steps that will guide you through creating a notebook and running a code in it.

Step 1: First, open the terminal and head to the directory where you want to save the notebook. Now, type the command Jupyter Notebook. The program will initiate a local server or at any other specific port.

Step 2: Once the Jupyter Notebook window pops up on your screen, create a new Notebook.

Step 3: On the editor window, type the following command:

```
print("Hello World!")
```

Step 4: Now, click on the cell where the code is written. Press SHIFT+ENTER. You can also click on the play button at the top of the cell.

Step 5: Once you run the code, the output will appear in the space at the bottom of the window. You can now press the stop button to stop running the code.

12. SUMMARY

In this introductory unit on Python, we've journeyed through the rich history and evolution of a programming language that has become a cornerstone in the field of software development. Starting from its inception in 1991 by Guido van Rossum, we explored Python's initial design to simplify programming tasks and its progression to a robust tool that seamlessly supports large-scale applications. The unit highlighted Python's unique position as both an interpreted and scripting language, emphasizing its rapid application development capabilities, ease of learning, and open-source nature.

We delved into the practical applications of Python, illustrating its widespread use in creating sophisticated web applications, driving advancements in machine learning and artificial intelligence, powering data science endeavours, and even in game development and business applications. The versatility of Python was showcased through its support for multiple frameworks, enhancing the security, scalability, and speed of web-based applications, and its critical role in the future of technology with libraries like NumPy and Pandas.

The unit also provided a comparative evaluation of Python with other languages, underscoring its accessible syntax, memory management, and extensive library support, which together make Python a preferred choice over languages like C, C++, Perl, and Tcl. Through this exploration, we aimed to equip you with a foundational understanding of Python, its features, and its unparalleled flexibility, setting the stage for deeper dives into programming concepts and applications in subsequent units.

13. GLOSSARY

- **Python:** A high-level, interpreted, and general-purpose programming language known for its simplicity, readability, and versatility in various applications.
- **Rapid Application Development (RAD):** A software development methodology that emphasizes quick prototyping and iterative delivery, characteristics supported by Python.
- **Interpreted Language:** A programming language in which most of its implementations execute instructions directly without previously compiling a program into machine-language instructions, unlike compiled languages.
- **Scripting Language:** A programming language used for writing scripts, which are programs written for a special runtime environment that automates the execution of tasks.
- **Open-Source Language:** Software for which the original source code is made freely available and may be redistributed and modified according to the requirement of the user.
- **Object-oriented Programming (OOP):** A programming paradigm based on the concept of "objects", which can contain data and code: data in the form of fields (often known as attributes or properties), and code, in the form of procedures (often known as methods).
- **Guido van Rossum:** The creator of Python, who began its development in the late 1980s and continues to play a significant role in its development.
- **CWl (Centrum Wiskunde & Informatica):** The national research institute for mathematics and computer science in the Netherlands, where Python was initially developed.
- **Python Software Foundation (PSF):** A non-profit organization devoted to the Python programming language, responsible for supporting its development and community.
- **Web Development:** The work involved in developing a website for the Internet or an intranet, ranging from simple single static pages to complex web-based applications.
- **Machine Learning:** A branch of artificial intelligence (AI) and computer science that focuses on the use of data and algorithms to imitate the way that humans learn, gradually improving its accuracy.

- **Data Science:** An interdisciplinary field that uses scientific methods, processes, algorithms, and systems to extract knowledge and insights from structured and unstructured data.
- **Game Development:** The process of designing, creating, and releasing a game. It involves concept generation, design, build, test, and release.
- **Business Applications:** Software designed to assist in solving specific business problems or to fulfil the needs of an organization.
- **Desktop GUI (Graphical User Interface):** A form of user interface that allows users to interact with electronic devices through graphical icons and visual indicators.
- **Web Scraping:** The process of using bots to extract content and data from a website.
- **Unicode:** A computing industry standard for the consistent encoding, representation, and handling of text expressed in most of the world's writing systems.
- **Tkinter:** Python's de-facto standard GUI (Graphical User Interface) package, used for creating simple GUI applications.

14. SELF-ASSESSMENT QUESTIONS

1. Python is known as:
   A) Only a high-level programming language
   B) Only a scripting language
   C) Both a high-level programming language and a scripting language
   D) A low-level programming language
2. Who developed Python?
   A) Linus Torvalds
   B) James Gosling
   C) Guido van Rossum
   D) Dennis Ritchie
3. Python was originally developed to:
   A) Replace Java
   B) Handle networking better
   C) Perform regular administrative tasks
   D) Enhance the capabilities of C++
4. Which of the following is NOT a real-life application of Python?
   A) Web Development
   B) Game Development
   C) Business Applications
   D) Hardware Programming
5. Python's functionality in web development is often enhanced by:
   A) Anti-virus software
   B) Web browsers
   C) Frameworks like Flask and Django
   D) Spreadsheet software
6. For data science, Python provides libraries such as:
   A) Adobe and Illustrator
   B) Pandas and NumPy
   C) Word and PowerPoint
   D) AutoCAD and SolidWorks

7. Python is often compared to which of the following languages due to its similar syntax?
   A) JavaScript
   B) PHP
   C) C and C++
   D) Ruby
8. Python provides various benefits over C and C++, such as:
   A) Faster compilation
   B) Memory allocation and management
   C) Lower-level access to hardware
   D) More complex syntax for advanced users
9. Python is slower than:
   A) Java
   B) Perl
   C) Tcl
   D) All the above
10. Which version of Python introduced Unicode support by default?
    A) Python 1. x
    B) Python 2. x
    C) Python 3. x
    D) Python 4. x
11. In Python 2, how is integer division handled by default?
    A) It returns a float
    B) It rounds up to the nearest integer
    C) It truncates to the nearest integer
    D) It prompts for user input
12. The transition from Python 2 to Python 3 was characterized by:
    A) Backward compatibility
    B) Incompatibility of some libraries
    C) No significant changes
    D) Decreased performance
13. Python is known for its:

A) Complex syntax like C++

B) Easy to understand and use nature

C) Requirement for extensive memory management

D) High dependency on third-party modules

14. Python's approach to memory management is:

A) Manual, requiring explicit commands

B) Managed by the Python interpreter

C) Identical to C and C++

D) Dependent on third-party packages

15. One of Python's major advantages for web development is its:

A) Limited library support

B) Rich library for various protocols like HTTP, FTP

C) Incompatibility with major web frameworks

D) Focus solely on desktop applications

15. TERMINAL QUESTIONS

1. Describe the unique features of Python that make it suitable for Rapid Application Development (RAD).
2. How does Python's dual role as both an interpreted and scripting language benefit developers?
3. Discuss the historical development of Python and its evolution from the ABC scripting language.
4. Explain how Python supports web development through various frameworks.
5. Describe the role of Python in Machine Learning and Artificial Intelligence with examples of libraries that support these technologies.
6. Discuss the significance of Python in Data Science and the libraries that facilitate data analysis and visualization.
7. Compare Python's syntax and usability with that of C and C++.
8. How does Python's memory management and allocation provide an advantage over other high-level languages?
9. Discuss Python's performance in comparison to Java and the development of JPython to integrate Python features with Java classes.
10. What are the key differences between Python 2 and Python 3, particularly in terms of Unicode support and integer division?
11. Why has Python 3 been considered the better choice for new programming projects over Python 2?
12. Discuss the challenges faced by developers during the transition from Python 2 to Python 3, especially regarding library compatibility.
13. Elaborate on the ease of use and simplicity of Python as a programming language.
14. How does Python's scalability make it a preferred choice for both small and large applications?
15. Discuss Python's object-oriented nature and how it supports both procedural and object-oriented programming without enforcing OOP concepts.
16. How does Python's interpreted nature contribute to its flexibility and ease of debugging?

17. Describe the benefits of Python being an open-source language and the role of the Python Software Foundation.
18. Explain how Python's extensive standard library supports web scraping applications and data handling.
19. Discuss the significance of Python's portability across different operating systems and its impact on cross-platform development.
20. How does Python's exception-handling mechanism enhance the reliability and maintainability of Python applications?

16. ANSWERS

Self-Assessment Questions

1. Answer: C) Both a high-level programming language and a scripting language
2. Answer: C) Guido van Rossum
3. Answer: C) Perform regular administrative tasks
4. Answer: D) Hardware Programming
5. Answer: C) Frameworks like Flask and Django
6. Answer: B) Pandas and NumPy
7. Answer: C) C and C++
8. Answer: B) Memory allocation and management
9. Answer: A) Java
10. Answer: C) Python 3. x
11. Answer: C) It truncates to the nearest integer
12. Answer: B) Incompatibility of some libraries
13. Answer: B) Easy to understand and use nature
14. Answer: B) Managed by the Python interpreter
15. Answer: B) Rich library for various protocols like HTTP, FTP

Terminal Questions

1. Refer to the discussion on Python's capabilities as a RAD tool and its features that support large application development.
2. Look into Python's flexibility being both an interpreted and scripting language and how this dual capability enhances development efficiency.
3. Explore the historical context provided around Python's inception, focusing on its roots in the ABC language and its evolution under Guido van Rossum.
4. Examine the portion detailing Python's utility in web development, particularly the mention of frameworks like Flask and Django.
5. Review the segments highlighting Python's contribution to Machine Learning and AI, paying attention to libraries like NumPy and Pandas.

6. Delve into the explanation of Python's role in data science, focusing on its data handling and visualization capabilities through various libraries.
7. Look at the comparison between Python and languages like C and C++, noting the syntactical and usability aspects.
8. Investigate the advantages of Python over other languages in terms of memory management and how the Python interpreter handles it.
9. Consider Python's performance relative to Java and the rationale behind the creation of JPython to leverage Java's features.
10. Focus on the differences between Python 2 and 3, especially regarding Unicode support and how integer division is handled.
11. Examine the reasons why Python 3 is recommended for new projects, considering its advancements and improvements over Python 2.
12. Study the challenges associated with transitioning from Python 2 to 3, with an emphasis on library compatibility issues.
13. Reflect on Python's design philosophy that emphasizes ease of use and readability, making it accessible to beginners and experts alike.
14. Contemplate Python's ability to scale from simple scripts to complex applications, and how it accommodates various programming paradigms.
15. Explore how Python supports both procedural and object-oriented programming, offering flexibility in application development.
16. Consider Python's interpreted nature and how it influences the development process, particularly in terms of flexibility and debugging.
17. Investigate the implications of Python being open-source and the contributions of the Python Software Foundation to its development.
18. Review the section on web scraping and data handling, focusing on Python's library support for these tasks.
19. Delve into Python's cross-platform capabilities, noting how its design as an ANSI C-compliant language enhances portability.
20. Examine Python's approach to exception handling, and how it aids in creating reliable and maintainable code.

17. SUGGESTED BOOKS AND E-REFERENCES

BOOKS:

- Eric Matthes (2016), Python Crash Course: A Hands-On, Project-Based Introduction to Programming.
- John M. Zelle (2009), Python Programming: An Introduction to Computer Science (Preliminary Second Edition).
- Mark Lutz (2011), Python Programming: A Powerful Object-Oriented Programming (Fourth Edition).
- Sebastian Raschka (2017), Python Machine Learning - Machine Learning and Deep Learning with Python (Edition 2)

E- REFERENCES:

- Python Programming Certification Training Course, last viewed on March 23, 2021, < https://www.edureka.co/python-programming-certification-training >
- Python Tutorials and Sample Programs, last viewed on March 23, 2021, < https://www.w3schools.com/python/ >
- History of Python, last viewed on March 23, 2021

  < https://en.wikipedia.org/wiki/History_of_Python >