PYTHON ESSENTIALS

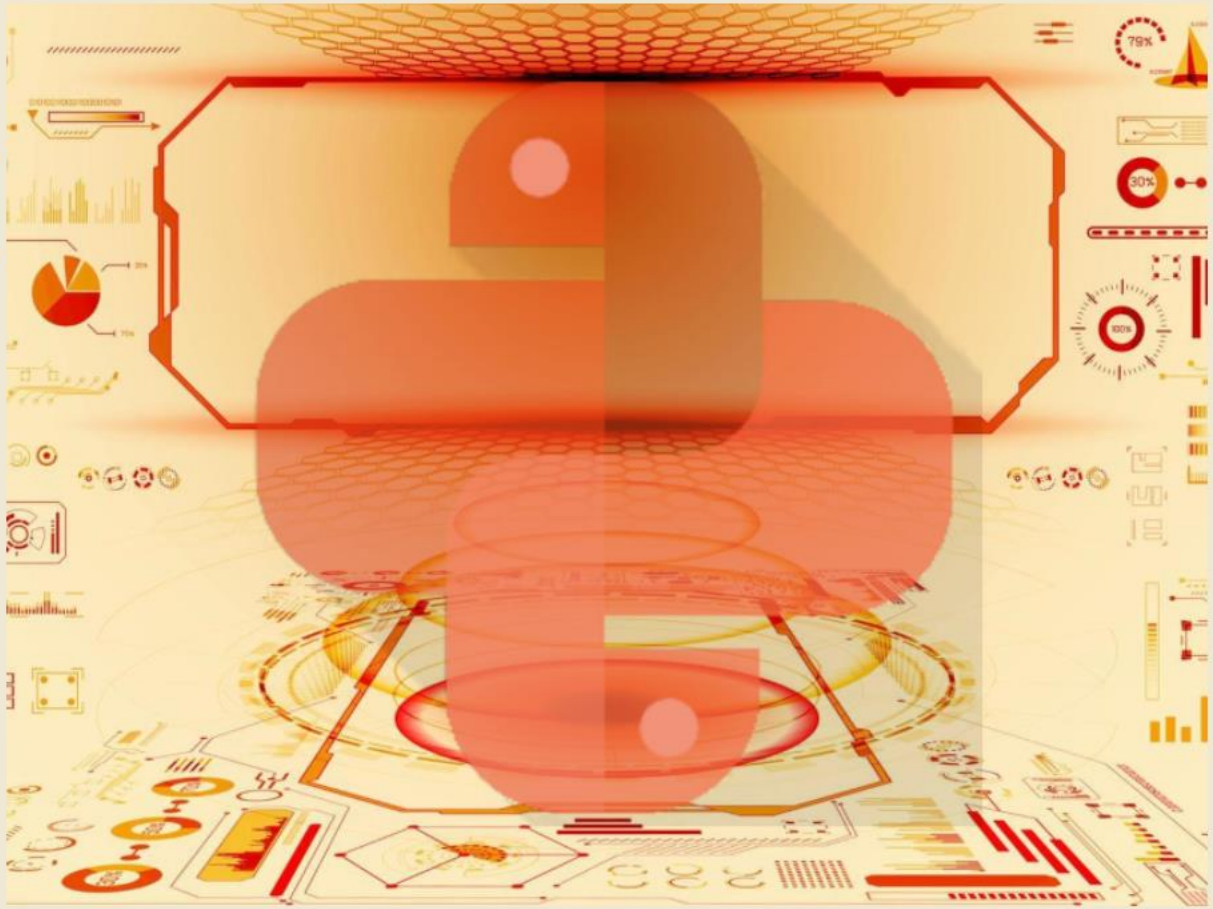
AN INTRODUCTION IN PROGRAMMING USING PYTHON

WHAT IS PYTHON?

UNDERSTANDING THE EXACT NATURE OF THE PYTHON LANGUAGE

From a general point of view Python is an interpreted, object-oriented, high-level programming language with dynamic semantics.

INTERPRETED	An interpreted language is a programming language in which its implementation execute instructions directly without earlier compiling a program into machine language.
OBJECT ORIENTED	In this case the language is centered around the concept of objects (encapsulation of data and functionality), rather than pure functionality (e.g. functional languages).
HIGH LEVEL	The programming language uses a strong abstraction level, being usually independent of the underlying computer architecture and operating system.
DYNAMIC SEMANTICS	It represents the fact the language variables may change their type and semantics dynamically, during the program's execution; the semantics is not preset directly in the source code.



IN DEPTH: INTERPRETED LANGUAGES CHARACTERISTICS

AN IN-DEPTH EXAMINATION OF INTERPRETED LANGUAGES IN COMPARISON WITH COMPILED LANGUAGES

An interpreted language means that the program runs by executing the source code line by line as the source code is processed by the interpreter.

	INTERPRETED LANGUAGES	COMPILED LANGUAGES
 SOURCE CODE EXECUTION	Source code is **executed line by line in real time** by the language interpreter. Therefore the **source code is executed in its original format** as it is processed by the interpreter.	Source code is **compiled before execution** by the language's compiler. The compiled artifacts are actually executed as a second step, after compilation.
 EXECUTION PERFORMANCE	As the code is **executed real time in its original format**, the speed of execution may be slower.	During compilation step the **performance of generated artifacts can be highly optimized**. As compilation generates machine format executables, the execution speed is further improved.
 DEVELOPMENT FLEXIBILITY	Due to the fact that the source code is executed real-time, there is **a lot of flexibility for software development** (including on-the-fly code changes, dynamic semantics or runtime introspection).	Since the code is statically compiled, there is **limited flexibility for software development**.
 DEVELOPMENT RELIABILITY	Development issues are **likely to be discovered later** (in most cases at execution time) due to the fact that the source code is executed real time.	Development issues are **likely to be discovered earlier** since the compilation process can detect these problems before compiled code execution.
 LANGUAGE APPLICABILITY	The flexibility and portability of interpreted languages ensure **fast development cycles**, making them suitable for Rapid Application Development and Proof of Concept implementations.	The compilation to machine language ensures a **high execution performance**, making compiled languages optimal for high performance, platform specific products.

In its standard implementation, Python source code is executed by a special software called **the Python virtual machine**. The source code is actually **converted at runtime to Python bytecode**, before being executed by the Python virtual machine. This is **not considered a compilation process**, as the source code is not translated directly into a platform specific binary format; it is the role of the Python virtual machine to translate the Python bytecode into machine specific instructions.

IN DEPTH: OBJECT ORIENTED LANGUAGES CHARACTERISTICS

AN IN-DEPTH EXAMINATION OF OBJECT ORIENTED LANGUAGES IN COMPARISON WITH LANGUAGES THAT DO NOT SUPPORT THIS PARADIGM

An object oriented language is a programming language having its development philosophy revolving around objects (both data and functionality) rather than just data (such as data manipulation languages) or just functionality (such as functional languages).

The most frequently utilized programming languages are either Object Oriented Languages (such a Java or C#) or Functional Languages (such as C or Lisp).

	OBJECT ORIENTED LANGUAGES	FUNCTIONAL LANGUAGES
 ENCAPSULATION	Data and functionality are **encapsulated in the same entity**. The access to data and operations is furthermore controlled via operations and **data access permissions**.	Data and functionality are **separated concepts**, sometimes with an unbalanced relevance. There is limited access control to data and operations, they are usually **accessible without restrictions**.
 REUSABILITY	The usage of objects was conceived for **maximal reusability** for both data and functionality. The same object can be **reusable simultaneously from data and functionality perspectives**.	The separation of data and functionality leads to a **complex mechanism for reusability**. Adding new functionality requires adding **both data and implementation in the same time**.
 APPLICATION STATE MANAGEMENT	Application state is **reflected through the state of its objects**, this state can be **changed through object method calls**. Therefore, object method calls may lead to **side effects**.	Application state is **reflected through its variables**. In general, function calls will **not produce side effects**; subsequent calls with the same inputs producing identical results.
 EXECUTION MODEL	The **order if execution is very important** since object method calls depends on and change the object state. The general programming model is close to the **imperative** paradigm.	Since functions are usually designed to be **independent of each other**, there is not mandatory order in which functions should be actually called - a **declarative** programming paradigm.

Python supports **both the object oriented and the functional software development paradigms**, making it a multi-paradigm software development language. This adds **even more flexibility to Python development**, you can program simple applications using functional programming and you can develop extensive functionalities leveraging the long term productivity of object oriented programming.
This makes Python a very attractive choice for **general purpose programming**, being suitable for a **large suite of applications** from desktop and web application to API implementations and machine learning.

IN DEPTH: HIGH LEVEL LANGUAGES CHARACTERISTICS

AN IN-DEPTH EXAMINATION OF HIGH LEVEL LANGUAGES IN COMPARISON WITH LOW LEVEL LANGUAGES

A high level programming language uses a high level of abstraction, obscuring machine specific details from the developers and allowing a high degree of expressivity and flexibility via natural-language like constructs.

	HIGH LEVEL LANGUAGES	LOW LEVEL LANGUAGES
 EXPRESSIVITY	These languages have a **strong similarity with natural language**. The language's reserved words are **similar with their natural language counterparts**, being highly expressive .	The language revolves closely around the **structure and technical architecture of a specific platform**. The language structure is usually less expressive.
 ABSTRACTION	Due to the highly expressive nature of these languages, they can **abstract concepts and entities from the business domain**, being able operate with complex models.	The platform specific, simplistic, nature of these languages lead to **very low capabilities for abstraction**. Such languages are usually used for narrow domain solutions which are platform specific.
 PORTABILITY	Due to their independence from a specific platform architecture these languages can create **highly portable applications**. The compiler or interpreter is specific to a target platform.	Since these languages are closely tied with a target platform, this **leads to creation of non-portable applications**, these applications need rewriting in case of a new platform.
 PRODUCTIVITY	The excellent expressivity, the capabilities for abstraction, reusability and portability **lead to a vey high boost to productivity**. Interpreted languages are increasing that boost.	The low expressivity, the low capabilities for abstraction and lack of portability **are significantly decreasing the productivity**. Lack of portability increases total cost of ownership.
 MAINTAINABILITY	The high abstraction capabilities, the separation of concerns, the expressive syntax and the modularity support lead to **implementation clarity** and **easier maintainability**.	Due to the low abstraction capabilities, the low expressivity and the almost total absence for lack of concerns separation,**the degree of maintainability is very limited**.

Python is a **high level programming language**, with **excellent abstraction and expressivity capabilities** along with strong **support for modularity** and mechanisms for implementing **separation of concerns**. Furthermore Python's syntax is **very terse** and **enforces a clear implementation structure**.
These features make Python complex applications **easier to maintain** and ensure a relatively **low cost for further implementation and customization**.

WHO IS USING PYTHON AND WHY?

LET'S UNDERSTAND WHICH COMPANIES ARE USING PYTHON AND WHY THE PYTHON LANGUAGE IS USED IN THESE COMPANIES

Google has been a supporter of Python since the early days.

In the beginning, they decided to use C++ only when memory control and low latency mattered. With everything else, they **chose Python for its ease of maintenance and fast delivery**. This strategy allowed them to maintain their code efficiently while also delivering faster.

The founders formulated the strategy as: **"Python where we can, C++ where we must"**.

In December of 2005, six months after the launch of the first version of Reddit, it was recoded into Python from Lisp.

The main reason for the change was that Python had more code libraries readily available and was more flexible. This allowed developers to add more functionality and provide quicker upgrades.

According to one of the company's founders: **"When we hire new employees ... I don't think we've yet hired an employee who knew Python. I just say, 'everything you write needs to be in Python.' Just so I can read it."**

Spotify, the world's largest music streaming service, has more than 280 million monthly users. This is the company using Python for data analysis and backend services.

Spotify leverages powerful Python to enhance Radio and Discover functions for users; approximately 80% of Spotify's backend services are written entirely in Python.

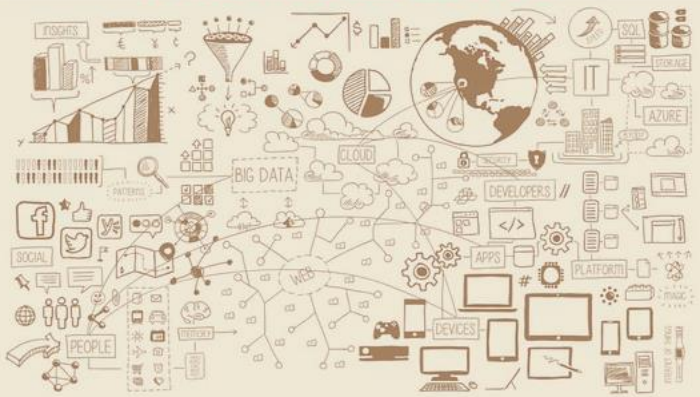
According to Spotify's engineering: **"Speed is a big focus for Spotify. Python fits well into this mindset, as it gets us big wins in speed of development."**

There are more than 100 leading companies that **have Python at the core of their application engineering**. They prefer Python for the **speed of development**, for its **high flexibility** and its **rich ecosystem of libraries**.
According to Enlyft data, in 2023 there are more than 200 000 companies using Python, most of them having between 10-50 employees and an yearly revenue between 1-10 million dollars. Most of these companies are in US, followed by companies in UK.
Consequently, Python is a great choice for **both mature companies and startups alike**, being the programming language that raises the highest interest in the IT industry.

ESSENTIAL USE CASES

EXPLORING THE MAIN USE CASES RELATED TO PYTHON LANGUAGE USAGE ACROSS IT INDUSTRY

MACHINE LEARNING



Python has become the **language of choice for many data scientists and machine learning practitioners** due to its clean syntax, extensive library support, and easy integration with other tools and libraries.
Python is able to perform **statistical and numerical computations efficiently** , including numerous built-in functions and libraries for **handling and processing data** , like NumPy and Pandas for storing and manipulating data.
Python is also widely used for **machine learning** thanks to its **excellent support for deep learning** through popular libraries like TensorFlow, Keras, and PyTorch.

WEB DEVELOPMENT

Python's popularity in web development is due to a number of factors, **including its extensive standard library**, available **web frameworks** , and ability to **integrate with other technologies** such as JavaScript and SQL.
Django is one of the **most popular web frameworks for Python**, it is easy to learn and use. It provides a range of features, making it easy to quickly develop web applications.
Jinja is a **powerful template engine**, providing a range of features including **dynamic tags and support for extensions**, making it very easy to generate web content.

SOFTWARE AUTOMATION

Python is a very good fit for software automation due to its **flexibility and multitude of integration libraries**.
Furthermore, Python has several mainstream frameworks for automating tasks and processes.
Ansible is a **popular deployment and configuration management tool** that makes it easy to automate tasks such as **software installation and configuration management** .
BotCity is an **advanced automation tool** used for both automated testing and RPA. Due to its large set of integration libraries, BotCity allows software automation for Web, desktop and mobile environments.

Python is present in many use cases related to the high impact IT industry sectors, being especially present in the **Data Science and Machine Learning** domain. Due to the large spectrum of integration libraries and frameworks, it has also a strong footprint in **Web Development and Software Automation**.
Python is however less present in the area of mobile development and gaming, which is more the domain of specialized or high performance languages.

INTERTECHNICA

BUSINESS SOLUTIONS

www.intertechnica.com
+40723065651
office@intertechnica.com

THANK YOU!