I. Introduction: =================

0. Introduce yourself.

1. \* Tell me about Python?

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together.

2. Why Python is so popular now a days?

Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance.

Python supports modules and packages, which encourages program modularity and code reuse.

The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

3. \* Features of Python:

1. Free and Open Source

[Python](https://www.geeksforgeeks.org/python-programming-language/)language is freely available at the official website

2. Easy to code

Python is a [high-level programming language](https://www.geeksforgeeks.org/difference-between-high-level-and-low-level-languages/). Python is very easy to learn the language as compared to other languages like C, C#, Javascript, Java, etc.

3. Easy to Read

Python’s syntax is really straightforward. The code block is defined by the indentations rather than by semicolons or brackets.

4. Object-Oriented Language

Python supports object-oriented language and concepts of classes, object encapsulation, etc.

5. GUI Programming Support

Graphical User interfaces can be made using a module such as [PyQt5](https://www.geeksforgeeks.org/pyqt5-qaction/), PyQt4, wxPython, or [Tk in Python](https://www.geeksforgeeks.org/python-gui-tkinter/). PyQt5 is the most popular option for creating graphical apps with Python.

6. High-Level Language

Python is a high-level language. When we write programs in Python, we do not need to remember the system architecture, nor do we need to manage the memory.

7. Large Community Support

Python has gained popularity over the years. Our questions are constantly answered by the enormous StackOverflow community. These websites have already provided answers to many questions about Python, so Python users can consult them as needed.

8. Easy to Debug

Excellent information for mistake tracing. You will be able to quickly identify and correct the majority of your program’s issues once you understand how to [interpret](https://www.geeksforgeeks.org/difference-between-compiled-and-interpreted-language/)Python’s error traces. Simply by glancing at the code, you can determine what it is designed to perform.

9. Python is a Portable language

Python language is also a portable language. For example, if we have Python code for Windows and if we want to run this code on other platforms such as [Linux](https://www.geeksforgeeks.org/introduction-to-linux-operating-system/), Unix, and Mac then we do not need to change it, we can run this code on any platform.

10. Python is an Integrated language

Python is also an Integrated language because we can easily integrate Python with other languages like C, [C++](http://www.geeksforgeeks.org/c-plus-plus/), etc.

11. Interpreted Language:

Python is an Interpreted Language because Python code is executed line by line at a time. like other languages C, C++, [Java](https://www.geeksforgeeks.org/java/), etc. there is no need to compile Python code this makes it easier to debug our code. The source code of Python is converted into an immediate form called bytecode.

12. Large Standard Library

Python has a large [standard library](https://www.geeksforgeeks.org/libraries-in-python/) that provides a rich set of modules and functions so you do not have to write your own code for every single thing. There are many libraries present in Python such as [regular expression](https://www.geeksforgeeks.org/regular-expression-python-examples-set-1/)s, [unit-testing](https://www.geeksforgeeks.org/unit-testing-software-testing/), web browsers, etc.

13. Dynamically Typed Language

Python is a dynamically-typed language. That means the type (for example- int, double, long, etc.) for a variable is decided at run time not in advance because of this feature we don’t need to specify the type of variable.

14. Frontend and backend development

With a new project py script, you can run and write Python codes in HTML with the help of some simple tags <py-script>, <py-env>, etc. This will help you do frontend development work in Python like javascript. Backend is the strong forte of Python it’s extensively used for this work cause of its frameworks like [Django](https://www.geeksforgeeks.org/django-tutorial/)and [Flask](https://www.geeksforgeeks.org/flask-creating-first-simple-application/).

15. Allocating Memory Dynamically

In Python, the variable data type does not need to be specified. The memory is automatically allocated to a variable at runtime when it is given a value. Developers do not need to write int y = 18 if the integer value 15 is set to y. You may just type y=18.

4. \* Advantages and Disadvantages of Python

Advantages of Python

1. Easy to Read, Learn and Write

Python is a**high-level programming language** that has English-like syntax. This makes it easier to read and understand the code.

2. Improved Productivity

Python is a very **productive language**. Due to the simplicity of Python, developers can focus on solving the problem. They don’t need to spend too much time in understanding the **syntax** or **behavior** of the programming language. You write less code and get more things done.

3. Interpreted Language

Python is an interpreted language which means that Python directly**executes the code** line by line. In case of any error, it stops further execution and reports back the error which has occurred.

Python shows only one error even if the program has multiple errors. This makes **debugging** easier*.*

4. Dynamically Typed

Python doesn’t know the type of variable until we run the code. It automatically assigns the data type during **execution**. The programmer doesn’t need to worry about declaring variables and their data types.

5. Free and Open-Source

Python comes under the **OSI approved** open-source license. This makes it **free**to **use** and **distribute**. You can download the source code, modify it and even distribute your version of Python. This is useful for organizations that want to modify some specific behavior and use their version for development.

6. Vast Libraries Support

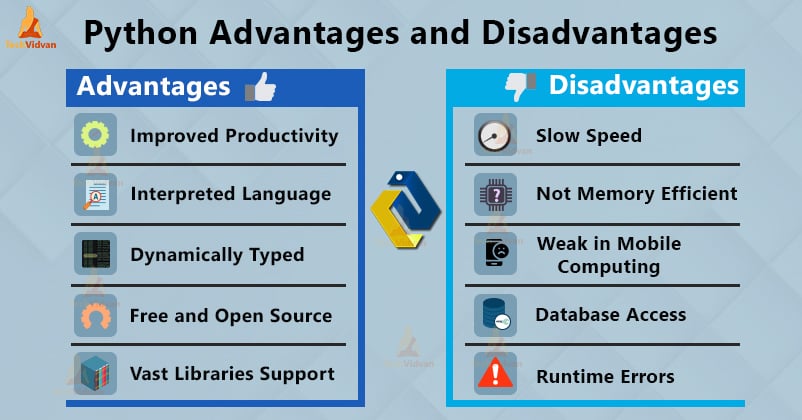
The standard library of Python is huge, you can find almost all the functions needed for your task. So, you don’t have to depend on external libraries.

But even if you do, a **Python package manager (pip)**makes things easier to import other great packages from the **Python package index (PyPi)**. It consists of over 200,000 packages.

7. Portability

In many languages like C/C++, you need to change your **code** to run the program on different platforms. That is not the same with Python. You only write once and run it anywhere.

However, you should be careful not to include any **system-dependent features**.

[](https://techvidvan.com/tutorials/wp-content/uploads/sites/2/2019/12/Python-advantages-disadvantages-1.jpg)

Disadvantages of Python

1. Slow Speed

We discussed above that Python is an **interpreted** language and **dynamically-typed** language. The line by line execution of code often leads to **slow execution**.

The dynamic nature of Python is also responsible for the **slow speed**of Python because it has to do the extra work while executing code. So, Python is not used for purposes where speed is an important aspect of the project.

2. Not Memory Efficient

To provide simplicity to the developer, Python has to do a little tradeoff. The Python programming language uses a **large amount of memory**. This can be a disadvantage while building applications when we prefer memory optimization.

3. Weak in Mobile Computing

Python is generally used in **server-side programming**. We don’t get to see Python on the client-side or mobile applications because of the following reasons. Python is **not memory efficient** and it has **slow processing power** as compared to other languages.

4. Database Access

Programming in Python is **easy** and **stress-free**. But when we are interacting with the database, it lacks behind.

5. Runtime Errors

As we know Python is a dynamically typed language so the data type of a variable can change anytime. A variable containing integer number may hold a string in the future, which can lead to **Runtime Errors**.

5. Interpreted vs Compiled time programming languages. Explain in detail

Programming languages can be broadly categorized into two types based on how they are processed: compiled and interpreted. Understanding the difference between the two can help you choose the right programming language for your project, and can also help you optimize your code for better performance. In this article, we’ll discuss the differences between compiled and interpreted programming languages.

Compiled Programming Languages

Compiled programming languages are those in which the source code is compiled into machine code before it is executed. Machine code is a low-level language that can be executed directly by the computer’s CPU. When a program written in a compiled language is compiled, the source code is converted into an executable file that can be run on the target machine. The compiled code is optimized for the specific hardware and operating system of the machine on which it is intended to run.

Some examples of compiled programming languages include C, C++, Java, and Swift. When you write code in a compiled language, you typically use an integrated development environment (IDE) to write and edit the code. The IDE will typically have a built-in compiler that can generate the executable file from the source code.

One advantage of compiled languages is that they tend to be faster than interpreted languages since the code is already translated into machine code before it is executed. However, this also means that compiled code is less portable since it is specific to the hardware and operating system of the machine on which it was compiled.

Interpreted Programming Languages

Interpreted programming languages, on the other hand, are those in which the source code is executed directly by an interpreter, without being compiled into machine code first. The interpreter reads the source code line by line and executes each line as it is read. The interpreter is responsible for translating the source code into machine code at runtime.

Some examples of interpreted programming languages include Python, Ruby, and JavaScript. When you write code in an interpreted language, you can typically run it directly from the command line or from an IDE without the need for a separate compilation step.

One advantage of interpreted languages is that they tend to be more portable than compiled languages, since the same source code can be executed on different hardware and operating systems without the need for recompilation. However, interpreted code is generally slower than compiled code, since it has to be translated into machine code at runtime.

Conclusion

In conclusion, the main difference between compiled and interpreted programming languages is in how they are processed. Compiled languages are translated into machine code before they are executed, while interpreted languages are translated into machine code at runtime by an interpreter. Both types of languages have their advantages and disadvantages, and the choice between them depends on the specific needs of your project. If you need maximum speed and performance, you may want to choose a compiled language, while if you need portability and ease of use, an interpreted language may be a better choice.

6. \* .py vs .pyc files

.py Files (Source Code Files)

These are the raw source code files written in Python programming language.

They contain human-readable code that is written by developers.

Python code in .py files is interpreted by the Python interpreter at runtime.

These files are used for development and are the files you write and edit to create your Python programs.

.pyc Files (Compiled Bytecode Files)

These files are generated by the Python interpreter from the corresponding .py source code files.

.pyc files contain compiled bytecode, which is a lower-level representation of the Python code.

The bytecode is generated to improve the execution speed of Python programs, as it is closer to machine code and can be executed faster by the interpreter.

.pyc files are automatically created and stored in the \_\_pycache\_\_ directory when you import a module for the first time.

Python uses .pyc files to avoid recompilation and speed up subsequent imports of the same module.

Conclusion

The .py files are the human-readable source code files written by developers, while .pyc files are the compiled bytecode files generated by the Python interpreter for improved execution speed and efficiency. The Python interpreter automatically creates and uses .pyc files to optimize the execution of your Python programs.

8. \* Why Python is Dynamically typed programming Language. Explain

*Python* is a dynamically typed language. What is dynamic? We don't have to declare the type of a variable or manage the memory while assigning a value to a variable in *Python*. Other languages like C, C++, Java, etc.., there is a strict declaration of variables before assigning values to them. We have to declare the kind of variable before assigning a value to it in the languages C, C++, Java, etc..,

*Python* don't have any problem even if we don't declare the type of variable. It states the kind of variable in the runtime of the program. *Python* also take cares of the memory management which is crucial in programming. So, *Python* is a dynamically typed language. Let's see one example.

## assigning a value to a variable

x = [1, 2, 3]

## x is a list here

print(type(x))

## reassigning a value to the 'x'

x = True

## x is a bool here

print(type(x))

## we can also redefine 'x' as many times as we want

<class 'list'>

<class 'bool'>

9. Python is Platform independent.Explain

Yes, Python is a platform-independent language. This means that the same Python code can run on different operating systems like Windows, MacOS, and Linux without any modifications. The Python interpreter takes care of the platform-specific details, allowing developers to focus on writing the code, rather than worrying about the underlying system. This makes it easier to develop and maintain Python applications, as the code can be reused across different platforms.

10.Different ways to write python program. Interactive, IDLE, CommandPrompt, IDE Advantages, Disadvantages

11.sourcecode vs bytecode

The primary distinction between source code and bytecode is that source code is a collection of computer instructions expressed in a human-readable programming language, whereas **bytecode** is an intermediate code that is processed by a **virtual machine** between **source code** and machine code.

When converting a programming language from source code to machine code, some languages transform the source code to **bytecode**, which is an intermediate code.

12.Register and instruction set

Instruction set commands are used when setting a register to a specific value, copying data from memory to a register or vice versa, and reading and writing data.

13.High Level vs Low level programming Language

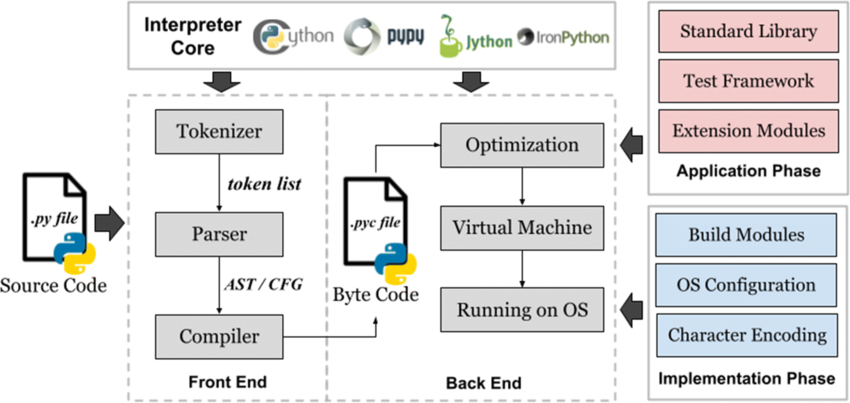
**High-Level Languages?**

* One can easily interpret and combine these languages as compared to the low-level languages.
* They are very easy to understand.
* Such languages are programmer-friendly.
* Debugging is not very difficult.
* They come with easy maintenance and are thus simple and manageable.
* One can easily run them on different platforms.
* They require a compiler/interpreter for translation into a machine code.
* A user can port them from one location to another.
* Such languages have a low efficiency of memory. So it consumes more memory than the low-level languages.
* They are very widely used and popular in today’s times.
* Java, C, C++, Python, etc., are a few examples of high-level languages.

**Low-Level Languages?**

* They are also called machine-level languages.
* Machines can easily understand it.
* High-level languages are very machine-friendly.
* Debugging them is very difficult.
* They are not very easy to understand.
* All the languages come with complex maintenance.
* They are not portable.
* These languages depend on machines. Thus, one can run it on various platforms.
* They always require assemblers for translating instructions.
* Low-level languages do not have a very wide application in today’s times.

14.\* Python architecture



15.\* Explain Garbage Collection mechanism in detail.

II. Variables: ===============

1. x = 10. Explain in detail for CRUD operations

2. tokens in Python. Explain all types

Python Tokens Types Python breaks each logical line into a sequence of elementary lexical components known as Tokens.

The normal token types are

1) Identifiers,

2) Keywords,

3) Operators,

4) Delimiters and

5) Literals.

**Identifiers**:

An Identifier is a name used to identify a variable, function, class, module or object. Example of valid identifiers Sum, total\_marks, regno, num1

**Keywords** :

Keywords are special words used by Python interpreter to recognize the structure of program. As these words have specific meaning for interpreter, they cannot be used for any other purpose.

**Operators** :

In computer programming languages operators are special symbols which represent computations, conditional matching etc. The value of an operator used is called operands.

Operators are categorized as Arithmetic, Relational, Logical, Assignment etc. Value and variables when used with operator are known as operands. Python supports following relational operators Python supports following Logical operators Python supports following Assignment operators Conditional operator Ternary operator is also known as conditional operator that evaluate something based on a condition being true or false. It simply allows testing a condition in a single line replacing the multiline if-else making the code compact. The Syntax conditional operator is, Variable Name = [on\_true] if [Test expression] else [on\_false] Example : min= 50 if 4950 else 70 # min = 70

**Delimiters:** Python uses the symbols and symbol combinations as delimiters in expressions, lists, dictionaries and strings. Following are the delimiters.

**Literals** :Literal is a raw data given in a variable or constant. In Python, there are various types of literals. 1) Numeric 2) String 3) Boolean 4)Escape Sequences In Python strings, the backslash "\" is a special character, also called the "escape" character. It

3. Garbage collection. How it works internally

4. Memory Management in Python

5. Dynamically typed programming. Explain examples.

6. Initializing variable, static,dynamic way

7. Assigning value to multiple variables. Explain

III. IDE PyCharm: ===================

1. Different IDEs in market

2. Advantages of IDE

3. Shortcuts in PyCharm (Explain min 10)

IV. Operators: ===============

1. Explain in detail about all operators

2. == vs is

3. and or operators. Explain

2 examples

4. Operator precedence.

5. Subtract 2 numbers and print result program. - Write down all scenarios with different input values V. DataTypes Intro: ==================== 1. Importance of DataTypes.2 2. Different data types, data structures available in Python 3. int vs float 4. Boolean. give all scenarios 5. 0 vs null

6. Explain each data type, data structure with real life examples.

7. CRUD Operations.Give examples for each

8. Sequences. Types of sequences. Sequence operations

9. Explain about below functions and give examples print() id() type() int() float() complex() bool() input() VI.

Keywords: =====================

1. Explain all keywords with examples and areas of usage