All About Python

Whether you're taking your first steps into the world of programming or brushing up on fundamental skills, mastering Python's basics is essential for building a strong foundation. Python, renowned for its readability and versatility, is an excellent choice for both beginners and seasoned developers alike.

This documentation aims to guide you through basic Python interview questions from understanding the history of Python to its uses, applications, versions, frameworks, interpreter and compiler working structures and functions. Each section is designed to be clear and accessible, providing you with practical examples and straightforward explanations.

Who Should Read This? If you're new to programming or looking to solidify your understanding of Python's fundamentals, this guide is tailored just for you. By the end, you'll have the tools you need to tackle more complex projects and continue your coding journey with confidence.

Dive in, explore, and start building your Python skills today!

Ver: 1

22/08/2024

1) What is Python? Its uses and Applications.

Python is a programming language that is interpreted, object-oriented, and considered to be high-level too. Python is one of the easiest yet most useful programming languages which is widely used in the software industry. People use Python for Competitive Programming, Web Development, and creating software. Due to its easiest syntax, it is recommended for beginners who are new to the software engineering field. Its demand is growing at a very rapid pace due to its vast use cases in Modern Technological fields like Data Science, Machine learning, and Automation Tasks. For many years now, it has been ranked among the top Programming languages.

Features of Python:

Python has plenty of features that make it the most demanding and popular. Let's read about a few of the best features that Python has:

- Easy to read and understand
- Interpreted language
- Object-oriented programming language
- Free and open-source
- Versatile and Extensible
- Multi-platform
- Hundreds of libraries and frameworks
- Flexible, supports GUI
- Dynamically typed
- Huge and active community

These also state the reasons why you should choose Python to learn as a beginner, or also to use it for development purposes as a developer, and a lot more.

Advantages and Disadvantages of Python:

Every programming language comes with benefits and limitations as well. These benefits and limitations can be treated as advantages and disadvantages. Python also has a few disadvantages over many advantages. Let's discuss each here:

Advantages of Python:

- Easy to learn, read, and understand
- Versatile and open-source
- Improves productivity
- Supports libraries
- Huge library
- Strong community
- Interpreted language

Disadvantages of Python:

- Restrictions in design
- Memory inefficient
- Weak mobile computing
- Runtime errors
- Slow execution speed

Uses and Applications of Python:

Python being so popular and so technologically advanced has multiple use cases and has real-life applications. Some of the most common Python applications which are very common are discussed below.

- **Web Development**

Developers prefer Python for web Development, due to its easy and feature-rich framework. They can create Dynamic websites with the best user experience using Python frameworks. Some of the frameworks are -Django, for Backend development and Flask, for Frontend development. Most internet companies, today are using Python framework as their core technology, because this is not only easy to implement but is highly scalable and efficient. Web development is one of the top Applications of Python, which is widely used across the industry to create highly efficient websites.

- **Data Science**

Data scientists can **build powerful AI models** using Python snippets. Due to its easily understandable feature, it allows developers to write complex algorithms. Data Science is used to create models and neural networks which can learn like human brains but are much faster than a single brain. It is used to extract patterns from past data and help organizations take their decisions. Also, companies use this field to make their future investments.

- **Web Scrapping and Automation**

You can also automate your tasks using Python with libraries like Numpy, Pandas, Matplotlib, etc. for **scraping and web automation.** Businesses use AI bots as customer support to cater to the needs of the customers, it not only saves their money but also proved to be providing a better customer experience. Web scrapping helps the business in analyzing their data and other competitors' data to increase their share in the market. It will help the organizations, make their data organize and scale business by finding patterns from the scrapped data.

- **CAD**

You can also use Python to work on **CAD (computer-aided designs) designs,** to create 2D and 3D models digitally. There is dedicated CAD software available in the market, but you can also develop CAD applications using Python also. You can develop a Python-based CAD application according to your customizability and complexity, depending on your project. Using Python for CAD development allows easy deployment and integration across cross-platforms.

- **Artificial Intelligence and Machine Learning**

Using libraries like Pandas, and TensorFlow, experts can work on **data analysis and machine learning applications** for statistical analysis, data manipulation, etc. Python is one of the most used Programming languages in this field. It is worth saying that Python is the language of AI and ML. Python has contributed a lot to this field with its huge collection of libraries and large community support. Also, the field of Artificial intelligence and Machine learning is exponentially evolving, hence the use of Python is also going to increase a lot.

- **Game Development**

Python can also be used by developers to **build games** using Pygame to develop 2D and 3D games. Some of the popular games built using Python are Pirates of the Caribbean, Battlefield 2, etc. Python has a library named ***Pygame***, which is used to build interesting games. Since the gaming industry is gaining a lot market in recent years the use of these kinds of development has increased a lot in recent past. Also, it is very easy to build games using this library, you can also try to build some basic games.

1) Why Python?

As stated earlier, the foremost reason for Python being so popular is its simplicity. Python doesn't require syntax technicalities; it uses natural language as its base. Programming in Python is as simple as typing an English sentence on your system. Python is also easy to download and install.
It is open-source, i.e. free to use and learn for everyone. Due to its ease of language, developers can actually spend most of their time learning useful programming skills instead of focusing on running the code. There are multiple reasons why Python is the most well-suited programming language for beginners: Reasons to Learn Python as a Beginner
There are numerous options for a beginner to choose a programming language in the world of computer science, so, why python? Let's discover 5 major reasons to learn Python as a beginner:

- **Easy to Learn**

The syntax of various other languages might seem very confusing to you when you just start coding. But, the minimal setup and readability of Python allow you to think like a programmer and save your time in writing a big syntax format. The syntax of this amazing language is easy and this makes it easier to type and compile. The diversity is so huge that **companies like Google, Netflix, and Spotify use Python**. This leverage of options is not available with other programming languages.

- **Readily Available Resources to Learn From**

Python is an **open-source language**, which is free and can be used by anyone. The internet is filled with **Python programming courses** and tutorials, which provide a broad scope of opportunities for beginners to actually hit and try to narrow down the niche in which they are better than others.
You can actually learn coding skills in just a matter of time..

- **The Extensibility Behavior of Python**

Extensibility in software and languages refers to the ability of the software to extend to include new additions with minimal to no change in the existing code. Simply put, it is the measure of how easily a new behavior can be added to its existing structure and code. Several programming languages like **Tcl** and Python, are highly extensible.

- **Python is Best Known for Its Versatility**

Python's versatility helps developers narrow down their niche and select one or two in which they are most confident. Python is a language that can be used in a range of different environments like **web development**, **data science**, **game development**, etc.

- **Python Community Allows You to Learn From the Experts**

Once your learning phase is over, and you start coding on your own, you might come across an obstacle or two. This is where the **Python community** will support you to learn from your own mistakes and learn from their experts indirectly.

2) What is the Framework?

A framework is like a structure that provides a base for the application development process. With the help of a framework, you can avoid writing everything from scratch. Frameworks provide a set of tools and elements that help in the speedy development process. It acts like a template that can be used and even modified to meet the project requirements.
Frameworks are based on programming languages. Some popular frameworks that are most used are Django, Flutter, Angular, Vue, PyTorch, Spring Boot, React Native, Apache Spark, Ionic, etc. These frameworks allow developers to create robust and rich functionalities software.

3) Why is Framework Used?

Writing the code from scratch is a tedious task full of possible risks and errors. You need to make the code clean, well-tested, bugs and errors free. It will be difficult for other developers to understand the code and work on it. So, it is better to work with the frameworks that meet your requirements. They make the development process easy with fewer errors. It is a general template that can be used and modified as per the requirement. It will be easy for others to understand your code as they are also familiar with frameworks.

Frameworks provide many advantages such as:

- Easy to test your code and debug it.
- Clean code is much easy to understand and work with.
- Reduces redundancy of code in the project.
- Reduces the time and cost of the project with the enhanced application.
- Features and functionalities provided by the framework can be modified and extended.

Examples: Frameworks are based on programming languages. Some popular frameworks that are most used are Django, Flutter, Angular, Vue, PyTorch, Spring Boot, React Native, Apache Spark, Ionic, etc.

4) History of python

- Python laid its foundation in the late 1980s.
- The implementation of Python was started in December 1989 by Guido Van Rossum at CWI in Netherland.
- In February 1991, Guido Van Rossum published the code (labeled version 0.9.0) to alt.sources.
- In 1994, Python 1.0 was released with new features like lambda, map, filter, and reduce.
- Python 2.0 added new features such as list comprehensions, garbage collection systems.
- On December 3, 2008, Python 3.0 (also called "Py3K") was released. It was designed to rectify the fundamental flaw of the language.
- ABC programming language is said to be the predecessor of Python language, which was capable of Exception Handling and interfacing with the Amoeba Operating System.
- The following programming languages influence Python:

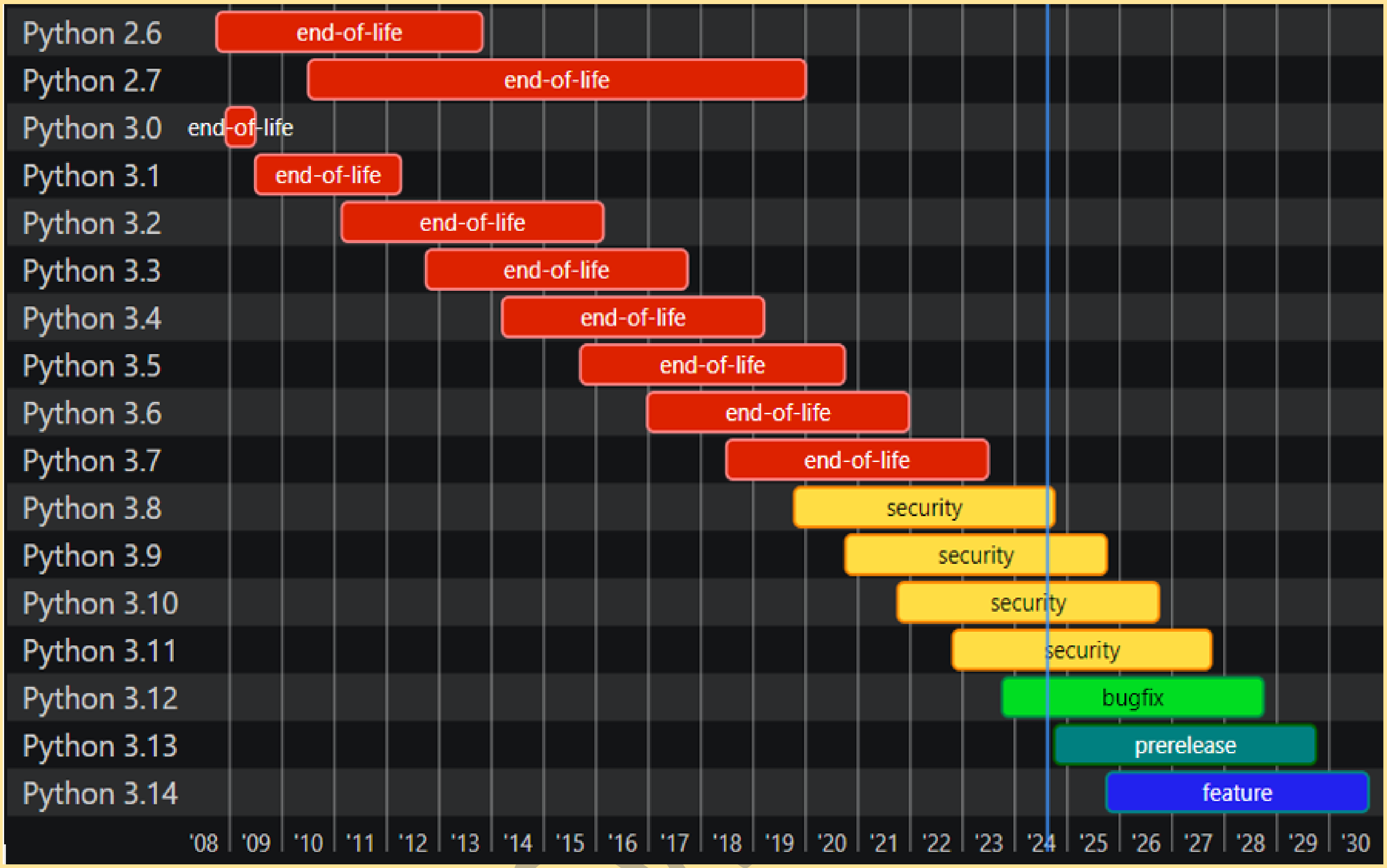
  a. ABC language.

  b. Modula-3

Status of Python versions

The main branch is currently the future Python 3.13, and is the only branch that accepts new features

Python release cycle:



5) Key Differences Between Python 2 and 3

- Print statement: In Python 2, the print statement is used as "print" whereas in Python 3 it is used as "print()".
- Division: In Python 2, division of integers results in a floor division (rounded down to nearest whole number) while in Python 3, division results in a float.
- Strings: In Python 2, strings are ASCII encoded by default while in Python 3 they are Unicode encoded.
- Exception handling: In Python 2, "as" is used as an alias for catching exceptions while in Python 3 "as" is used as a keyword.
- Integer types: Python 3 removed the distinction between long integers and regular integers, and they are now simply referred to as integers.
- Range function: In Python 2, range() generates a list, whereas in Python 3 it generates an object of type range, which is iterable.
- Metaclasses: In Python 2, you need to use the "metaclass" argument when defining a class, whereas in Python 3 this is done using the "metaclass" keyword argument in class definition.
- Raise statement: In Python 2, the "raise" statement is used to raise an exception, whereas in Python 3 it is used as an expression to raise exceptions.
- Input method: In Python 2, the "input" method takes input as a string, while in Python 3 it takes it as an expression.
- Unicode: In Python 2, there are separate functions for encoding and decoding Unicode, whereas in Python 3 these functions have been combined into a single method "encode".

6) What is Python interpreter?

Python is an interpreted language developed by Guido van Rossum in the year of 1991. As we all know Python is one of the most high-level languages used today because of its massive versatility and portable library & framework features. It is an interpreted language because it executes line-by-line instructions. There are actually two ways to execute python code one is in Interactive mode and another thing is having Python prompts which is also called script mode. Python does not convert high level code into low level code as many other programming languages do rather it will scan the entire code into something called bytecode. every time when Python developer runs the code and start to execute the compilation part execute first and then it generates a byte code which get converted by PVM Python Virtual machine that understand the analogy and give the desired output.

Interpreted Languages: Perl, BASIC, Python, JavaScript, Ruby, PHP.
Compiled Languages: C, C++, C#, COBOL and CLEO.

7) What is the Difference Between a Python Interpreter and a Compiler?

Computer programs are generally written in a high-level language, also called source code. Since machines don't understand this type of language, the language needs to be transformed into binary or machine code, which machines understand.

Compilers and interpreters make this transformation possible. However, while they both have the primary function of changing source code into machine code, there are differences between them.

These two remarkable tools significantly differ in how they translate the source code.

Interpreters translate source code one statement at a time. On the other hand, the compiler first scans the entire program and then translates the whole program into machine code.

These two methods of translating code present unique opportunities and challenges. Let's consider these.

- ✓ Interpreters translate programs one statement at a time, unlike compilers that do "batch translation." Therefore, interpreters usually take less time to analyze the source code. However, while they analyze code faster, compilers execute the code faster than interpreters.
- ✓ Secondly, because interpreters don't generate any Object code, they are more memory efficient than compilers.

8) Are Compilers Better than Interpreters?

It depends on what you want. Programs with interpreters can run right away and be started faster. Furthermore, interpreters make debugging your code easier since they identify line-by-line errors. Any changes made to the code written via a compiler will require it to be converted. Compilers no longer use computing power, making them more efficient.

9) How does an Interpreter work in python?

The Python interpreter is CPython and is written in the C programming language. So how does CPython work? Let's see.

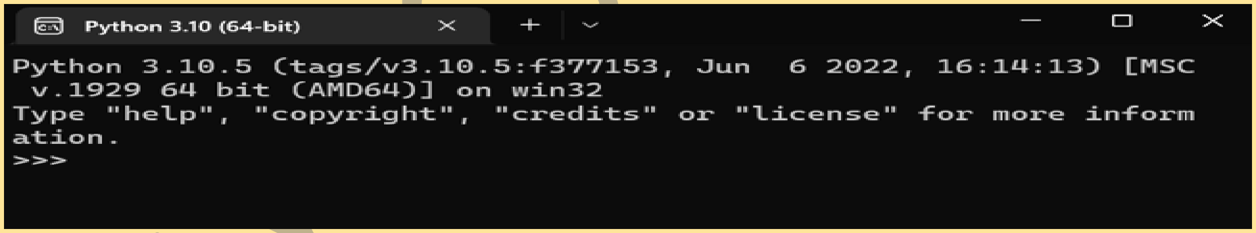
- **Interpreters start with source code analysis.**
  First, the interpreter checks or analyzes the source code. By now, we assume you know what the source code means. It's the high-level language you write programs in.So, CPython receives the source code and initializes several commands to do some vital things. First, the interpreter ensures you follow Python's syntax for writing code. It also checks for incorrect lines of code. If it encounters any error in a line, it stops the program from running and produces an error message. This analysis divides the source code files into a list of tokens.

- **The interpreter then generates byte code**.
  After lexical analysis, which is the process described in the section above, the interpreter moves to the second stage, byte code generation. After receiving the tokens, the interpreter generates the Abstract Syntax Tree or AST. This tree is converted to machine language (i.e., 1s and 0s). Because this is a Python interpreter, the code can be saved in a file with the **.pyc** extension. Next

- **The interpreter initializes a PVM**.
  The interpreter initializes the Python Virtual Machine (PVM) in the next step. PVM is crucial because it's what converts the byte code into binary code. After conversion, the results are then printed. Python prints out the correct result if there are no high-level language errors. Otherwise, it prints out an error message. Now that we have a high-level view of how the compiler works, let's now show you how to set up and use a Python interpreter.

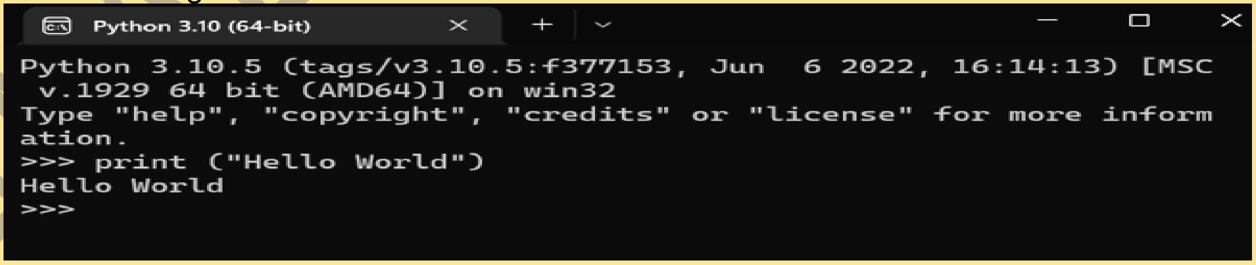
10) How to Run a Python Interpreter?

After downloading and installing Python, you can then run the Python interpreter. In our example, we have installed the installer on a Windows computer. Here's how to run the interpreter. First, search for the Python program on your computer. Opening it should show you something similar to what we have in the image below.



```
Python 3.10.5 (tags/v3.10.5:f377153, Jun  6 2022, 16:14:13) [MSC
 v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more inform
ation.
>>>
```

Now enter a command or statement. The interpreter will provide the appropriate results. We've decided to go for the tested and trusted "Hello World." You can see the results below.



```
Python 3.10.5 (tags/v3.10.5:f377153, Jun  6 2022, 16:14:13) [MSC
 v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more inform
ation.
>>> print ("Hello World")
Hello World
>>>
```

The basic Python interpreter allows you to execute single statements. However, if you want to execute multiple statements or build Python applications, you'll need more. This is where Integrated Development Environment (IDE) comes in. These applications give programmers extensive software capabilities like editing source code, building executables, and debugging.

Some of the best IDEs we recommend include:

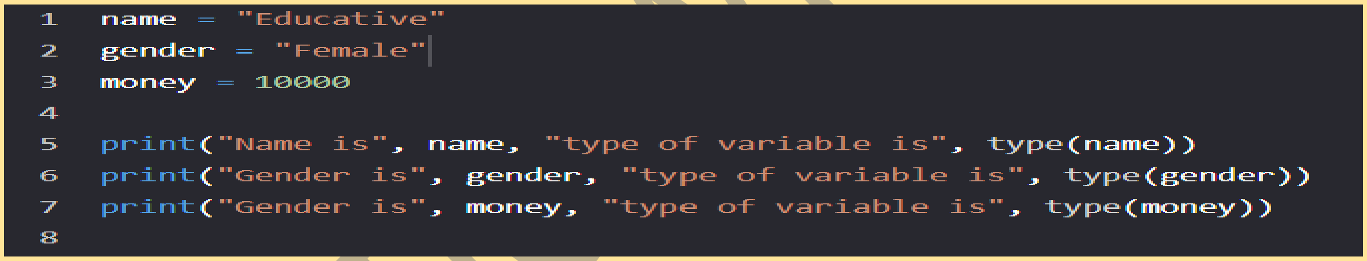
- Jupyter
- Spyder
- Visual Studio Code
- Sublime Text 4
- Atom

Installing these applications is relatively easy. All you have to do is visit their websites, download, and install. Hope you have a better understanding of what a Python interpreter is. First, it converts source code into machine code via tokenization. These tokens are then used to create an AST (Abstract Syntax Tree). Finally, the AST is converted to byte code, and the PVM executes the byte code to give the final output. Using an interpreter vital for programming and is one of the easiest steps to learning Python.

11) Why is Python a Dynamically Typed Language?

Python is dynamically typed because its variable types are decided upon and verified while the program is running. Therefore, type inference takes place automatically without requiring explicit type declarations. Variables may be changed from one type to another while a program runs, thanks to the dynamic typing feature, which offers flexibility and expressiveness. It facilitates quick prototyping and makes writing code easier but also necessitates careful planning to prevent potential runtime problems. Python’s dynamic typing finds a balance between practicality and security, and it can be used in conjunction with the optional usage of static typing via type hints for improved type verification.

Let’s look at the code below better understand what dynamic typing is.



```python
name = "Educative"
gender = "Female"
money = 10000

print("Name is", name, "type of variable is", type(name))
print("Gender is", gender, "type of variable is", type(gender))
print("Gender is", money, "type of variable is", type(money))
```

12) Difference Between Compile Time and Run Time in Python?

- What is compile-time?

Compile-time is the time period when a program code is translated into a low-level code or machine code, either by a compiler or an interpreter. Compile-time is the period of time from the beginning to the end of the process.

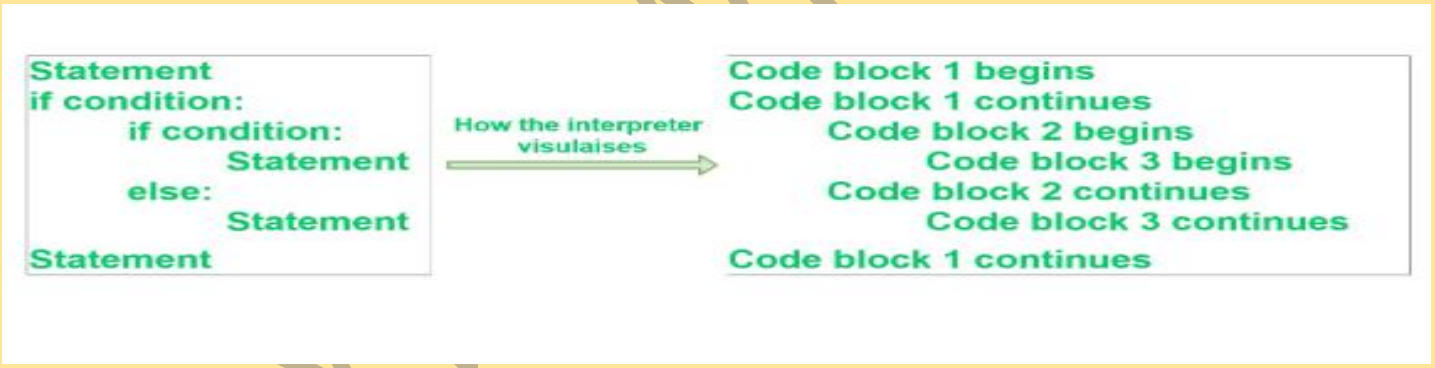
- What is runtime?

Runtime refers to the time when the converted code, which is now in low level or machine code, is executed. In other words, runtime refers to the time when the co.de does what it was written to do.

Runtime	Compile-time
The time period during code execution.	The time period during code translation.
Errors caught here will cause abnormally or to terminate unexpectedly. Fatal errors in PHP would do this.	Most of your program errors are caught here. For example, syntax, warning, and notice errors in PHP.
Cares less about object and function information as it doesn't know them, and only carries out the action associated with them.	Proper understanding of functions and objects, and assigns needed properties.
The codes are in the form of zeros and ones or a very similar one.	Codes are in human readable forms.
Errors caught here are usually not noticed during development and debugging.	Errors caught here can be easily fixed and corrected.
What comes at the end of the runtime is the result of computation or action performed by the computer, whatever it may be.	The output is a machine code that is understandable by computers.

13) Why is Indentation Important in Python?

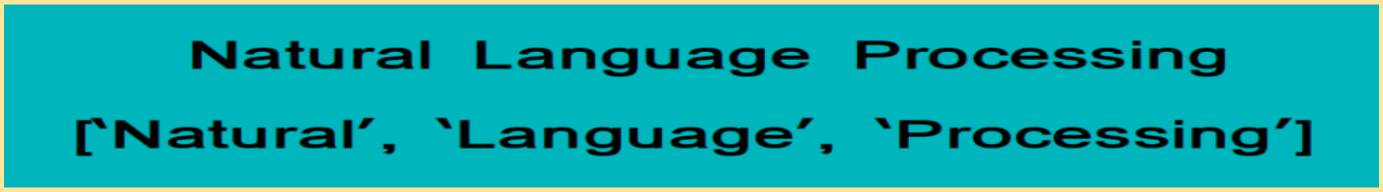
Indentation is a very important concept of Python because without properly indenting the Python code, you will end up seeing Indentation Error and the code will not get compiled. Python indentation refers to adding white space before a statement to a particular block of code. In other words, all the statements with the same space to the left, belong to the same code block.



Python uses indentation to highlight the blocks of code. Whitespace is used for indentation in Python. All statements with the same distance to the left belong to the same block of code. If a block has to be more deeply nested, it is simply indented further to the left. You can understand it better by looking at the following lines of code.

14) What is Tokenization

Tokenization in Python is essentially splitting a phrase, sentence, paragraph, or an entire text document into smaller units, such as individual words or terms. Each of these smaller units are called tokens.


Natural Language Processing
['Natural', 'Language', 'Processing']

The tokens could be words, numbers or punctuation marks. In tokenization, smaller units are created by locating word boundaries. Wait – what are word boundaries? These are the ending point of a word and the beginning of the next word. These tokens are considered as a first step for stemming and lemmatization.

Three simple types of tokenization in Python:

- ➢ Word Tokenization: Splitting a sentence into individual words.
- ➢ Sentence Tokenization: Breaking a paragraph into separate sentences.
- ➢ Regular Expression Tokenization: Using patterns to split text based on specific rules or conditions.

15) What Is the Difference Between Bytecode and Binary Code?

Byte code is a platform-independent intermediate code generated during compilation, typically associated with languages like Python. It's not directly executable by hardware. Binary code, however, consists of machine-specific instructions represented in 0s and 1s, directly interpretable by the CPU.

16) What Is the difference between Bytecode and Compiled Code?

Byte code serves as an intermediary between source code and machine code, designed for execution by a virtual machine. Compiled code refers to source code translated directly into machine code specific to a particular hardware architecture.

17) What Is the Difference Between Bytecode and Object Code?

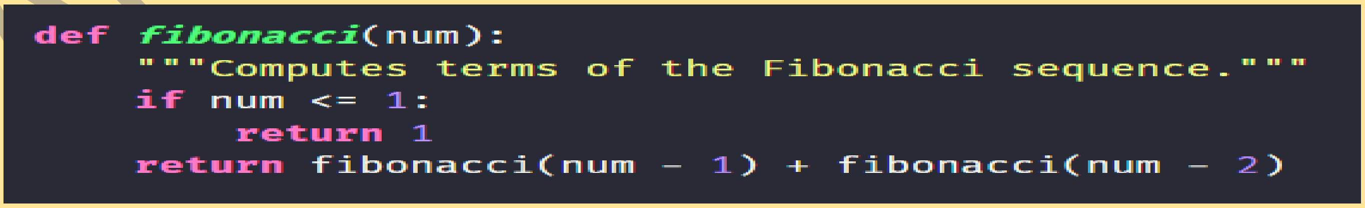
Bytecode is a platform-independent intermediate representation of code. Object code, however, is machine-specific and represents the compiled version of source or object-oriented programming languages like Python, C++ and Java.

18) What Is the Difference Between Bytecode and Source code?

Bytecode is a low-level representation of code generated from the compilation of high-level programming languages. Source code, in contrast, is the human-readable form of a program written by developers before compilation.

19) How to View Bytecode in Python?

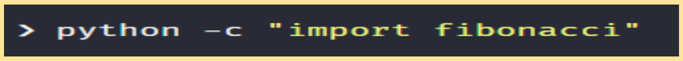
The dis module supports the analysis of CPython bytecode by disassembling it.If you have a .pyc file, you can use the modules dis and marshal from the standard library to get the corresponding bytecode: Suppose that you have a file fibonacci.py with the following code:



```python
def fibonacci(num):
    """Computes terms of the Fibonacci sequence."""
    if num <= 1:
        return 1
    return fibonacci(num - 1) + fibonacci(num - 2)
```

If you import your function fibonacci into the REPL or from another file, Python will compile the bytecode and write it to a .pyc file.

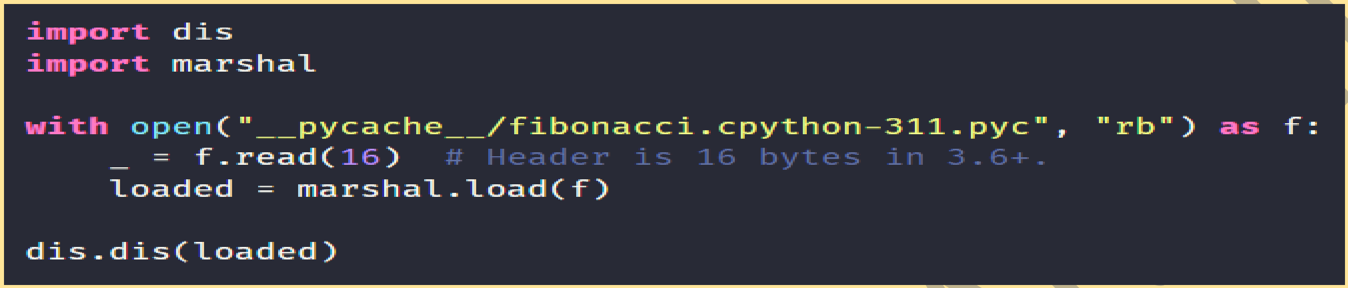
The quickest way to force Python to compile and dump the bytecode in a file is with this command:



```
> python -c "import fibonacci"
```

This will create a folder __pycache__ (if it doesn't exist yet) and write the bytecode to a file. Because I'm running Python 3.11 at the time of writing, the file that I got was fibonacci.cpython-311.pyc.

Now, to get the bytecode back from that file, I run the code from above, but I specify the path to the file fibonacci.cpython-311.pyc:

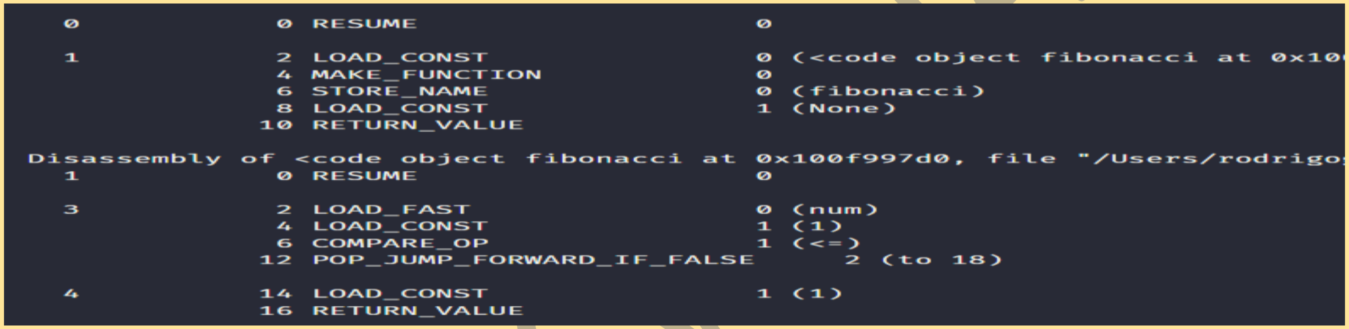


```python
import dis
import marshal

with open("__pycache__/fibonacci.cpython-311.pyc", "rb") as f:
    _ = f.read(16)  # Header is 16 bytes in 3.6+.
    loaded = marshal.load(f)

dis.dis(loaded)
```

If you run the code above, you get the bytecode associated with the .pyc file you opened:



```
  0           0 RESUME                   0

  1           2 LOAD_CONST               0 (<code object fibonacci at 0x10
              4 MAKE_FUNCTION            0
              6 STORE_NAME               0 (fibonacci)
              8 LOAD_CONST               1 (None)
             10 RETURN_VALUE

Disassembly of <code object fibonacci at 0x100f997d0, file "/Users/rodrigo
  1           0 RESUME                   0

  3           2 LOAD_FAST                0 (num)
              4 LOAD_CONST               1 (1)
              6 COMPARE_OP               1 (<=)
             12 POP_JUMP_FORWARD_IF_FALSE     2 (to 18)

  4          14 LOAD_CONST               1 (1)
             16 RETURN_VALUE
```

20) Can We Execute Bytecode in Linux?

Bytecode is an instruction set designed for efficient execution by a software interpreter or a virtual machine. Unlike machine code, which is directly executed by the CPU, bytecode is an intermediate representation of code, sitting between the high-level source code written by programmers and the machine code executed by the computer hardware. One of the key features of bytecode is its platform independence. This means that it can run on any device or operating system that has a compatible virtual machine, making the same bytecode executable across different platforms without modification.

21) Is Python Interpreted language or Compiled Language or Both?

Between compiled and interpreted languages, Python blurs the distinction. Because it can be compiled and interpreted, developers have a unique degree of flexibility. Let's look at Python's dual nature implementation. Python can be both compiled and interpreted, therefore when we run a Python program, it is first compiled and then line-by-line interpreted. When we say Python is both compiled and interpreted, it means that Python code is first compiled into an intermediate bytecode, which is then executed by the Python interpreter. This compilation step transforms human-readable code into a form that can be executed by the interpreter. This bytecode is platform-independent, allowing Python programs to run on any system with a compatible interpreter. Python is an interpreted language, which means the source code of a Python program is converted into bytecode that is then executed by the Python virtual machine. Python is different from major compiled languages, such as C and C++, as Python code is not required to be built and linked like code for these languages.

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

In this example, the Python interpreter converts the print statement into bytecode and executes it, displaying "Hello, World!" on the screen.

22) What are the most useful tools in Python?

Development tools help us to build fast and reliable Python solutions. It includes Integrated Development Environment (IDE), Python package manager, and productive extensions. These tools have made it easy to test the software, debug, and deploy solutions in production.

➢ **Jupyter Notebook**

Jupyter Notebook is a web-based IDE for experimenting with code and displaying the results. It is fairly popular among data scientists and machine learning practitioners. It allows them to run and test small sets of code and view results instead of running the whole file. The Jupyter Notebook lets us add a description and heading using markdown and export the result in the form of PDF and .ipynb files.

➢ **Pip**

Pip is a tool that uses Python Package Index to install and manage Python software. There are 393,343 projects for you to download and install with lightning speed. The Python ecosystem works on it.

pip install <package_name>

Pip is not just an installer. You can create and manage Python environments, install dependencies, and install packages from third-party repositories using URLs.

➢ **VSCode**

Visual Studio Code is free, lightweight, and a powerful code editor. You can build, test, deploy, and maintain all types of applications without leaving the software window. It comes with syntax highlighting, code auto-completing, language, Git, and in-line debug support. You can use extensions to pre-build systems and deploy applications to the cloud. VSCode is the most popular IDE in the world, and its popularity is mainly due to free extensions that improve user experience.

Conclusion:

In summary, understanding the basics of Python is crucial for anyone looking to delve into programming. Whether it’s grasping fundamental concepts like variables, data types, or control structures, or simply getting Comfortable with syntax, these building blocks set the stage for more advanced learning.

Thanks to my beloved mentor Anand Jah Sir for the wonderful guidance.

For more Basics to Advanced Interview Questions please follow me on:

LinkedIn

GitHub