# Python interview questions

| | |
|---|---|
| 1. What is Python and what are its features? | • **Free and open source** —<br>  ◦ The Python language is freely available to download on the official website.<br>  ◦ It is also open-source, meaning that the source code is publicly available. This makes it open to public contribution, as well as transparent in respect to security.<br>• **High level language, high readability and writability** —<br>  ◦ Python is a high-level programming language (as opposed to a low level language) — this means that the language has a strong abstraction from the details of the computer or from machine language. It uses natural language elements that make it easier to understand and developer-friendly. High level languages generally have higher readability, writability, and portability compared to low level languages.<br>  ▪ High level languages also may often automate significant areas of the computing systems, such as memory management. Rather than dealing with things like registers, memory addresses, and call stacks, high level languages prioritize a focus on usability over optimal program efficiency when implementing their concepts and components, such as: variables, arrays, objects, arithmetic, boolean expressions, subroutines, functions, loops, threads, locks, and other abstract computer science concepts.<br>• **Object-Oriented language** — Python supports object oriented programming and its concepts.<br>  ◦ object-oriented programming is a programming paradigm that uses objects and classes in programming. It aims to implement real-world entities like inheritance, polymorphisms, encapsulation, etc. in the programming. The main concept of OOPs is to bind the data and the functions that work on that together as a single unit so that no other part of the code can access this data.<br>• **Dynamically Typed Language, and Dynamic Memory Allocation** — Python is a dynamically-typed language, meaning that the data types for variables (such as int, double, long) are decided at runtime. This goes for the memory allocation of the variables as well. For this reason, we do not have to specify the data type of variables when writing python code.<br>• **Python is an Interpreted Language** — code is executed line by line at a time, in contrast to languages like Java, C, and C++, which need to be compiled. Python code is converted into its bytecode via interpretation rather than compilation. This makes the code easier to debug.<br>• **Portability, Bytecode** — Python source code is interpreted and converted into bytecode, which is then run by the Python Virtual Machine. This means that Python code is portable and can easily be run across platforms such as Linux/Unix/Mac/Windows.<br>• **Integrated language and Extensible feature** — Python is an integrated and extensible language. Some Python code can be written and compiled into C/C++, as well as integrated into other languages like Java, Rust, and Go.<br>• **Easy to debug** — Strong information for mistake tracing via Python's error traces.<br>• **Large standard library** — Python has a large standard library that provides a rich set of modules and functions. There are many libraries present for areas such as regular expressions, unit testing, web browsers, and more.<br>• **GUI programming support** — Python has modules for aiding in creating GUIs, such as PyQt5, PyQt4, wxPython, and tkinter |

| | |
|---|---|
| | • **Frontend and Backend Development** — with PyScript, Python code can be embedded or linked inside HTML to be executed inside the browser. This allows Python to be utilized for frontend development. Python can also be used for backend development with the help of frameworks like Django and Flask |

2. Explain the difference between Python 2 and Python 3.

- Some of the notable difference with the introduction of Python 3 are as follows in the following categories:

| | Python 2 | Python 3 |
|---|---|---|
| Syntax | Compared to Python 3, syntax is generally more complicated | Python 3 syntax is simpler and easier to understand compared to Python 2 |
| Storage of Strings | Strings are stored as ASCII by default and have to be marked with a "u" prefix | Strings are stored as Unicode by default |
| "print" keyword | Print in considered to be a statement and not a function | Print is considered to be a function and not a statement |
| Division of Integers | When two integers are divided, you are always provided an integer value (i.e. 3/2 = 1, rather than 1.5) | When two integers are divided, you are always provided a float value |
| Variable leakage | The value of global variables will change when using inside for-loops | The value of variables never changes |
| Rules of Ordering Comparisons | Rules of ordering comparison are very complex | Rules of ordering comparisons are simplified compared to Python 2 |
| Exceptions | Exceptions are enclosed in notations | Exceptions are enclosed in parenthesis |
| Iteration | The xrange() function is used for iterations | The range() function is used for iterations |
| Libraries | A lot of libraries for Python 2 are note forward compatible | A lot of libraries created for Python 3 are strictly used for Python 3 |

| | |
|---|---|
| | |

| | | |
|---|---|---|
| Backwards Compatibility | Python 2 can be ported to Python 3 but is high effort and unreliable | Python 3 is not backwards compatible with Python 2 |
| | | |

| | |
|---|---|
| 3. What are the built-in data types in Python? | Python has a handful of data types that fall under a span of categories: |

| | |
|---|---|
| Text Type: | str |
| Numeric Types: | int, float, complex |
| Sequence Types: | list, tuple, range |
| Mapping Type: | dict |
| Set Types: | set, frozenset |
| Boolean Type: | bool |
| Binary Types: | bytes, bytearray, memoryview |
| None Type: | NoneType |

| | |
|---|---|
| 4. What is the difference between a tuple and a list in Python? | • The primary difference between tuples and lists in Python is that tuples are immutable, while lists are mutable. |

| | |
|---|---|
| | • This means that the contents of a list can be modified, but the contents of a tuple cannot change after being created. |

| | |
|---|---|
| 5. How do you create a dictionary in Python? | • A dictionary is the built in mapping data type and can be created either with curly braces, {}, or by using the dict() constructor.<br>• The dict() constructor can used in several forms for taking arguments:<br>  • class dict(**kwarg)<br>  • class dict(mapping, **kwarg)<br>  • class dict(iterable, **kwarg) |

| | |
|---|---|
| 6. What is a generator in Python? | • Generators are functions that return a generator object, which is an iterable set of items that can be iterated through one at a time.<br>• Generator functions contrast with regular functions in that they use the *yield* keyword instead of the *return* keyword<br>• A *yield* statement pauses the execution of the function it is used in and returns the value to the caller, but maintains the status of the function to allow the function to continue after execution from where it left off. When resumed, the function continues execution immediately after the last run of *yield*. This generates a set of values over time, rather than computing them at once and returning them as a list.<br>• Generator expressions generate values in a "just in time" manner, which is more memory efficient and better optimized for high-performance scenarios such as complex datasets<br>• Generators are useful when we want to produce a large sequence of values, but we don't want to store them in memory all at once<br>• Generator objects use "lazy evaluation" to yield sequences, and call the *next()* method on the generator object to iterate through the values |

| | |
|---|---|
| 7. What are the different types of loops in Python and when would you use each one? | ● The 2 types of loops in Python are a for-loop and a while-loop<br>● A nested loop may be considered a 3rd type of loop, but it still consist entirely of either a for-loop or a while-loop. The only difference of a nested loop is the implication that the loop is nested inside another loop, i.e. it is enclosed by an outer loop.<br>● A for loop is used to iterate over a sequence in Python.<br>  ● It is more like the for-each loop in Java, in that it does not involve instantiating an indexing variable to reference the individual elements of the data structure. Rather, it just creates a variable to be the placeholder for each element in the data structure as it iterates.<br>  ● The for loop will iterate for the number of iterations as there are elements in the sequence it is passed<br>  ● Despite for loops being more like for-each loops (of other languages) by default, index variables can still be derived from for loops via unpacking<br>● e.g. *for index, x in lst*<br>● A while loop is used to execute a set of statements as long as the given condition is True<br>  ● A *condition* is given for the while loop. If the *condition* evaluate to True, the code statements indented in the while block will be executed and the control will be returned to the *condition* again to be checked. If the *condition* evaluates to False, the the while block statements will not be executed and control flow will move to right past the while block.<br>● A nested loop is used when an iterable sequence of steps are needed to be executed within each iteration of another sequence. |

| | |
|---|---|
| 8. What is a decorator in Python? | • A decorator is a function that takes another function and extends the behavior of the latter function without explicitly modifying it.<br>• Decorators are basically like wrapper functions for other functions.<br>• Decorators provide a simple syntax for calling higher-order functions<br>  ◦ Example:<br><br>`# Python program to illustrate functions`<br>`# can be passed as arguments to other functions`<br>`def shout(text):`<br>`  return text.upper()`<br><br>`def whisper(text):`<br>`  return text.lower()`<br><br>`def greet(func):`<br>`  # storing the function in a variable`<br>`  greeting = func("""Hi, I am created by a function passed as an argument.""")`<br>`  print (greeting)`<br><br>`greet(shout)`<br>`greet(whisper)`<br><br>• Output:<br><br>HI, I AM CREATED BY A FUNCTION PASSED AS AN ARGUMENT.<br>hi, i am created by a function passed as an argument. |

| | |
|---|---|
| 9. How do you handle exceptions in Python? | • Exceptions are caught and handled by using the *try* and *except* keywords in Python. The keywords are followed by a colon (:) and indented blocks that pertain to each.<br>• The code to be attempted is placed in the try block. Should an exception arise in the execution of the try-block code, a matching except block will be checked for to handle the exception. If a matching exception block is found, the code inside the except block is executed.<br>• We can use the *raise* keyword to throw an exception<br>• The *else* keyword can be used with try-except blocks, as a means to run a block of code, should no exceptions arise during the try block, i.e. the *except* block is not executed<br>• The *finally* keyword can be used to execute blocks of code that should always run, regardless of whether or not an exception arose during the *try* block. |

| | |
|---|---|
| 10. What is the difference between a module and a package in Python? | • A module is a simple Python file, with the extension *.py*, which can contain collections of functions and global variables.<br>• A package is a simple directory containing a collection of modules<br>  ◦ A package contains a *__init__.py* file which is is used by the Python interpreter to interpret it as a package.<br>  ◦ Packages can contain subpackages within them.<br>  ◦ *Numpy* and *Pandas* are examples of Packages |

| | |
|---|---|
| 11. What is PEP 8 and why is it important? | • PEP8 is a document that provides guidelines and best practices for writing Python code, and was written in 2001.<br>• The primary focus of PEP8 is to improve the readability and consistency of Python code<br>• PEP8 is important because it exists to improve the readability of Python code and help overall collaboration with others and understanding within the Python community.<br>• PEP stands for Python Enhancement Proposal and there are several PEPs. A PEP is a document that describes new features proposed for Python and documents aspects of Python, like Design and style for the community.<br>• Some of the topics covered in PEP8 are:<br>  ◦ Naming conventions<br>  ◦ Code layout<br>  ◦ Indentation<br>  ◦ Comments<br>  ◦ Whitespace in Expressions and Statements<br>  ◦ Programming Recommendations |

| | |
|---|---|
| 12. How do you debug Python code? | • Python has the *pdb* module (Python Debugger) which comes as a standard built-in library for debugging Python code. *Pdb* internally makes use of the *bdb* module (basic debugger functions) and *cmd* (support for line-oriented command interpreters)<br>• The major advantage of pdb is it runs purely in the command line, making it great for debugging code on remote servers that don't have a GUI-based debugger |

13. How would you reverse a string in Python?

| | |
|---|---|
| 14. What are the advantages of using NumPy over regular Python lists? | • NumPy arrays are faster and more compact compared to to Python lists, consuming less memory ot store data<br>• NumPy is the fundamental package for scientific computing in Python, and NumPy *array* is the core data structure of the package.<br>• The NumPy arrays provide a wide variety of mathematical operations and support creating *n*-dimensional arrays of homogeneous data<br>• Unlike Python lists, NumPy arrays are homogeneous and only allow a single data type to be stored.<br>• It provides a mechanism for for specifying the data types |

15. What are the different types of inheritance in Python?

16. How do you implement a linked list in Python?

| | |
|---|---|
| 17. What is the difference between deep and shallow copying in Python? | • The difference between shallow and deep copy in Python in only relevant for compound objects (i.e. objects that contain other objects, like lists or class instances)<br>• A shallow copy constructs a new compound object and then inserts references into it to the objects found in the original<br>• A deep copy constructs a new compound object and then recursively inserts copies (i.e. not references to the original objects) into it of the objects found in the original. |

18. What are the benefits of using a virtual environment in Python?

| | |
|---|---|
| 19. Explain the difference between *args and **kwargs in Python. | • *args and **kwargs are used to pass a variable number of arguments to a function. They are used when we are unsure about the number of arguments to pass into the functions.<br>• *args is used to pass a variable number of non-keyworded arguments to a function<br>• **kwargs is used to pass a variable number of keyword arguments to a function through a dictionary, on which dictionary operations can be performed |

20. How do you check if a string is a palindrome in Python?