**20 Python Interview Questions To Challenge Your Knowledge**

## A peek into data structures, programming concepts, and best practices

While continuously learning about the ins and the outs of my favorite programming language, I always keep track of the interesting stuff across multiple Notion pages.

In this article, I turned some of my notes into**20 interview questions that cover data structures, core programming concepts, and Python best practices.**Interestingly, a lot of these questions are asked in data science interviews.

Hope you’ll go through some of them and brush up on your Python skills.

Without much further ado, let’s jump right in. 🏊

**[Join Medium with my referral link - Ahmed Besbes](https://medium.com/membership/@ahmedbesbes" \t "_blank)**

[Read every story from Ahmed Besbes (and thousands of other writers on Medium). Your membership fee directly supports…](https://medium.com/membership/@ahmedbesbes" \t "_blank)

[medium.com](https://medium.com/membership/@ahmedbesbes" \t "_blank)

**1— What is the difference between a list and a tuple? When should you use each?**

A list is a mutable data structure while a tuple is an immutable one.   
A mutable object in Python has the ability to change its values.

Lists are dynamic: you can add items to them or override and remove existing ones.   
Tuples are fixed-size: they don’t have an appendor an extendmethod. You cannot remove items from them either.   
Both tuples and lists support indexing and allow using the inoperator to check for existing elements in them.

**→ There are some situations where I think tuples might be useful.**

* If you declare a collection of items that you know will
* never change or that you will loop over only without changing its values, use tuples.
* If you look for performance, tuples are faster than lists since they’re read-only structures. If you don’t need write operations, consider using tuples.
* Tuples can make your code safer if you want to prevent accidentally writing data that doesn’t need to be changed.

Here’s a code sample that show how tuples differ from lists.

**2 — What is the difference between multiprocessing and multithreading? When should you use each?**

Multiprocessing and Multithreading are programming paradigms that aim to speed up your code.

When you use multiprocessing, you parallelize your computation over processes. Processes are independent and don’t communicate with each other: they don’t share the same memory area and have strict isolation between. In terms of applications, multiprocessing is suited for CPU-intensive workloads. It does, however, have a large memory footprint that is proportional to the number of processes.

On the other hand, in multithreaded applications, threads live inside a single process. Consequently, they share the same memory area: they can modify the same variables and can interfere with one another. While processes are strictly executed in parallel, only one thread is executed at a given point in time in Python, and this is due to the Global Interpreter Lock ([GIL](http://wiki.python.org/moin/GlobalInterpreterLock)). Multithreading is suited to IO-bound applications such as web scraping or fetching data from a database.

→ If you want to learn more about multithreading and multiprocessing, I recommend you go through this amazing blog [post](https://medium.com/contentsquare-engineering-blog/multithreading-vs-multiprocessing-in-python-ece023ad55a) that draws a comprehensive picture of the two concepts.

**3 — What is the difference between a module, a package, and a library?**

A module is simply a Python file that’s intended to be imported into scripts or other modules. It contains functions, classes, and global variables.

A package is a collection of modules that are grouped together inside a folder to provide consistent functionality. Packages can be imported just like modules. They usually have an \_\_init\_\_.pyfile in them that tells the Python interpreter to process them as such.

A library is a collection of packages.

**4 — What is the problem with multi-threading in python?**

The Global Interpreter Lock (or GIL) prevents the python interpreter from executing more than one thread at the same time. Put simply, the GIL forces that only one thread is executed at any point in time in Python.

This represents a big performance bottleneck in CPU-bound applications that rely on multithreaded code.

**5 — What are decorators? Can you describe a situation in which decorators are worth using?**

A decorator is a function that receives a function as input and returns a function as output. The goal of a decorator is to extend the behavior of the input function without changing its core mechanism.

Using a decorator also prevents you from repeating yourself. It forces you to write a generic code once and then tap it to every function that needs it.

A typical use-case where decorators shine is **logging**.

Imagine, for example, that you want to log to the terminal all the values of the parameters that are passed to every function that is called in your program. You can go through every function definition and write that down or you can just write one single decorator that does this logging task and apply it to all the functions that need it.

Applying a decorator to a function is only a matter of adding a single line above that function’s definition.

**# without decorator def my\_awesome\_function():   
 # do awesome stuff # with a decorator @my\_awesome\_decorator   
def my\_awesome\_function():   
 # do even more awesome stuff**

Here’s a code sample that creates a decorator called log that logs the values of the parameters that are passed to a function.

Decorators can also be used for other purposes such as timing functions, validating input data, enforcing access control and authentication, caching, etc.

**6 — How to properly write data to a file? What can go wrong otherwise?**

*Using a context manager is key.*

When you use the open statement without a context manager and some exception occurs before you close the file (closing the file is something you must remember when opening a file this way) memory issues could happen and the file might be corrupted along the way.

When you use withto open a file and an exception occurs, Python guarantees that the file is closed.

**7 — Are function arguments passed by reference or by value?**

All function arguments are passed by reference in Python: this means that if you pass a parameter to a function, the function gets a reference to that same object.

If the object is mutable and the function changes it, the parameter will mutate in the outer scope of the function. Let’s see an example:

**8 — How to override the way objects are printed?**

Use the \_\_str\_\_ and the \_\_repr\_\_ dunder methods.

Here’s an example that demonstrates how an instance from the Person class can be nicely formatted when printed to the console.

**9 — Write a function that computes the factorial of an integer n**

*Recursivity is key*

**10 — What is the difference between the is and == operators?**

== is an operator that tests the equality while is is an operator that tests for identity.

Two objects can have equal values without necessarily being identical (i.e. having the same memory address).

Remember that a is b is syntactic sugar for id(a) == id(b) .

**11 — When shouldn't you use the assert statement?**

The assert statement is useful for internal testing and sanity checks.

However, it shouldn’t be used to perform data validation or error handling because it’s generally disabled in production code for performance reasons.

Imagine if you check for admin privileges using assert: this can introduce a big security leak in production.

Instead of using the assert statement, you can throw a custom error.

**12 — What is a Python generator?**

A Python generator is a function that produces a sequence of items.

Generators look like typical functions but their behavior is different. For starters, instead of using the return statement, they use the yield statement.

Then, calling the generator function doesn’t run the function: it only creates a generator object. The code of the generator only executes when the nextfunction is applied to the generator object or if the generator is iterated over (in this case, the next function is implicitly called)

The number of times the next function is called on the generator object is equal to the number of times the yield statement is invoked in the generator function.

You can define generators using for-loops or generator expressions.

**13 — What is the difference between a class method and a static method? When should you use which?**

<https://cosasdedevs.com/posts/metodos-clase-metodos-estaticos-python/>

A static method is a method that knows anything about the class or the instance that had called it. It’s a method that logically belongs to the class but doesn’t have implicit arguments.   
A static method can be either called on the class or any of its instances.

A class method is a method that gets passed the class it was called on, much like selfis passed to other instance methods in a class.**The mandatory argument of a class method isn’t a class instance: it’s actually the class itself.**

A typical use-case of class methods is providing an alternative way to construct instances: a class method that does this is known as a *factory of the class*.

Here’s an Employee class that uses a class method that creates an instance in a slightly different way than the main constructor of the class.

**14— Give an example of how you use zip and enumerate**

The zip function takes multiple iterables as input and aggregates them in a tuple. For example, this can be useful if you want to loop over two lists at the same time.

The enumerate function allows to loop over an iterable and access both the running index and the item at the same time.

**15 — How would you use \*args and \*\*kwargs in a given function?**

\*args and \*\*kwargs make Python functions more flexible by accepting a variable number of arguments.

* \*args pass a variable number of non-keyworded arguments in a list
* \*\*kwargs pass a variable number of keyword arguments in a dictionary

Here’s an example of a function that takes a variable number of keyworded arguments that are collected in a dictionary called data (note that it doesn’t need to be named kwargs )



Image by the author — A function that uses \*\*kwargs

**16 — Give an example of functional programming using map**

**17 — what is the difference between the continue and break statements**

The break statement terminates the loop that contains it. The program immediately moves to the code section that is in the outer scope of the loop.

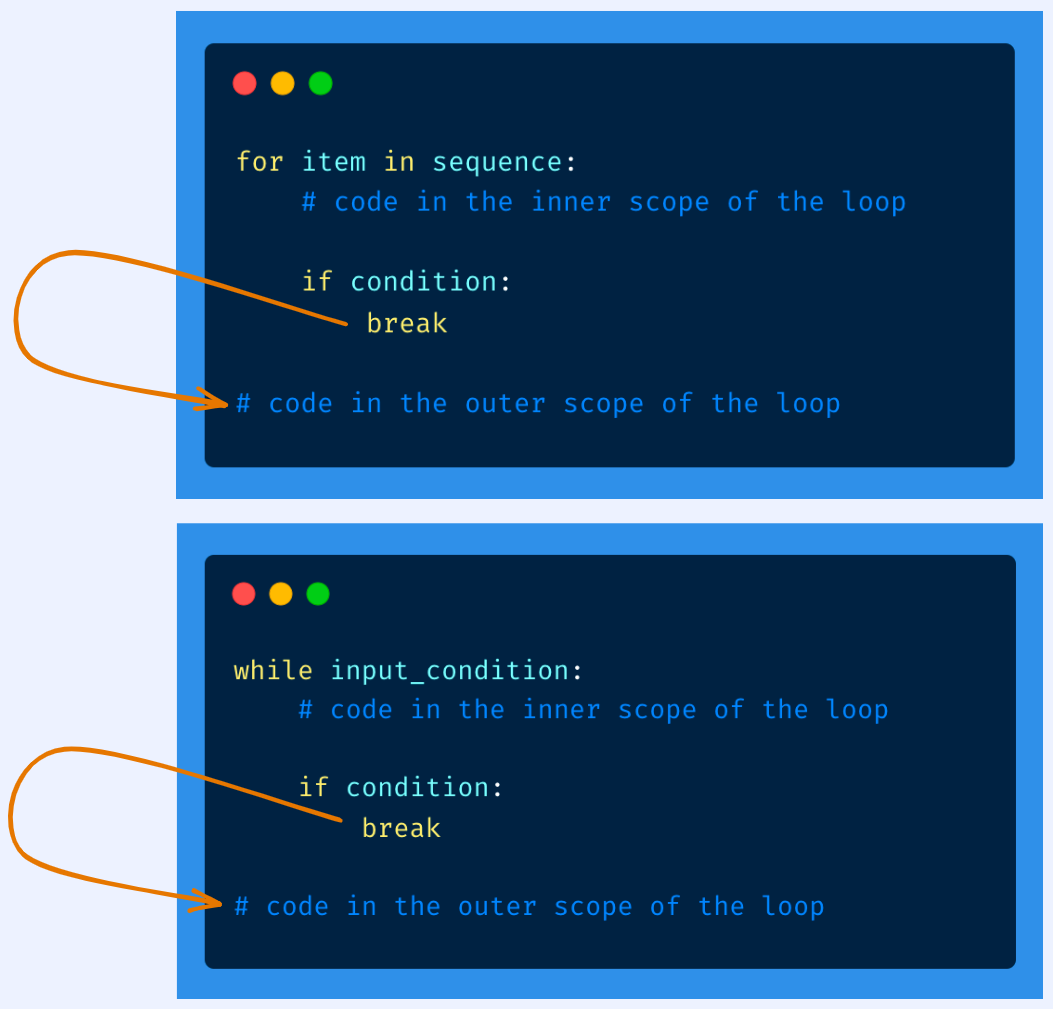


Image by the author

On the other hand, the continue statement skips the rest of the code of the current iteration and move to the next iteration.

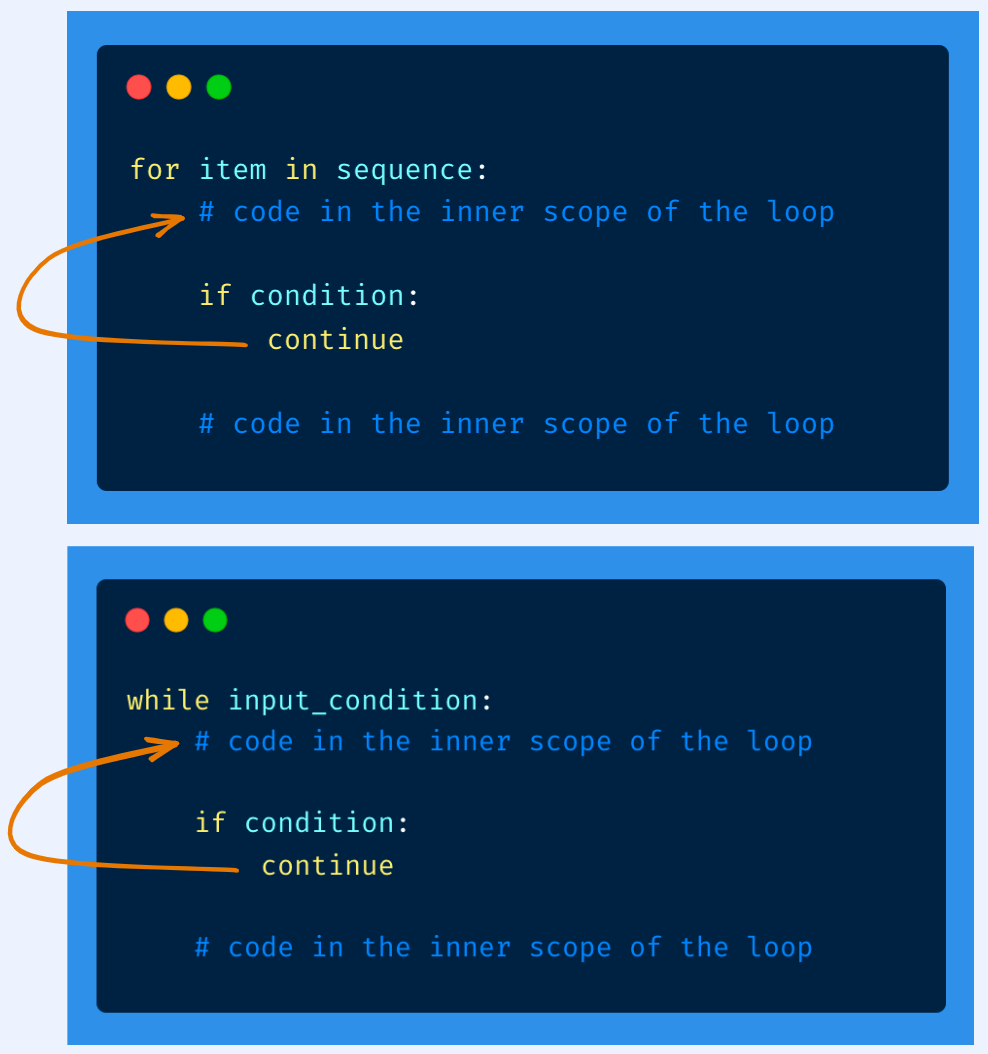


Image by the author

**18 — How to prevent a function from being called an unnecessary amount of time?**

*Use caching.*

If the output that is associated with a given input doesn’t change over a period of time, using caching would make sense for the function.

A typical scenario would be querying a web server: if you query a URL the first time and you know that response won’t change, you can cache the result.

**19 — Give some PEP8 guidelines**

* Use 4 spaces per indentation level.
* Imports should be grouped in the following order:

1. Standard library imports.
2. Related third-party imports.
3. Local application/library-specific imports.

* Function and variable names should be lowercase and separated by underscores
* Class names use the CapWords convention.

**20 — How to read an 8GB file in Python with a computer that has 2GB of RAM?**

This solution works for any large (and even larger) files.

When you open the file, all you need to do is use the file object as an iterator: while looping over this file object, you’ll be fetching one line at a time and the previous lines will be cleared from memory (i.e. they are garbage collected).

This way, the file will never be entirely loaded in memory and your processing will be done on the go.

**References 📖**

* <https://www.analytixlabs.co.in/blog/python-interview-questions-for-data-science/>
* <http://wiki.python.org/moin/GlobalInterpreterLock>
* <https://stackoverflow.com/questions/986006/how-do-i-pass-a-variable-by-reference>
* <https://medium.com/contentsquare-engineering-blog/multithreading-vs-multiprocessing-in-python-ece023ad55a>
* <https://peps.python.org/pep-0008/>
* <https://stackabuse.com/pythons-classmethod-and-staticmethod-explained/>