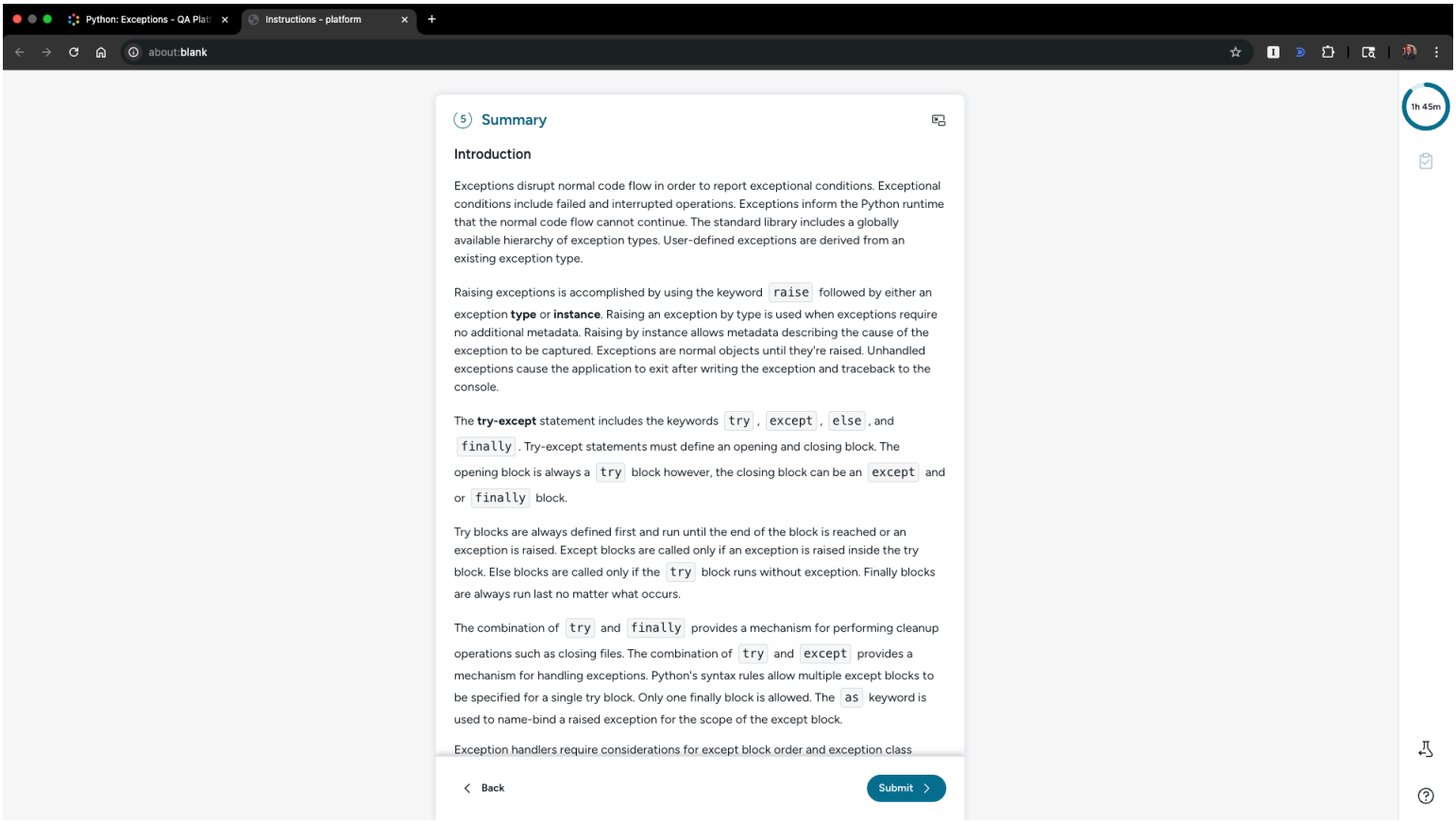
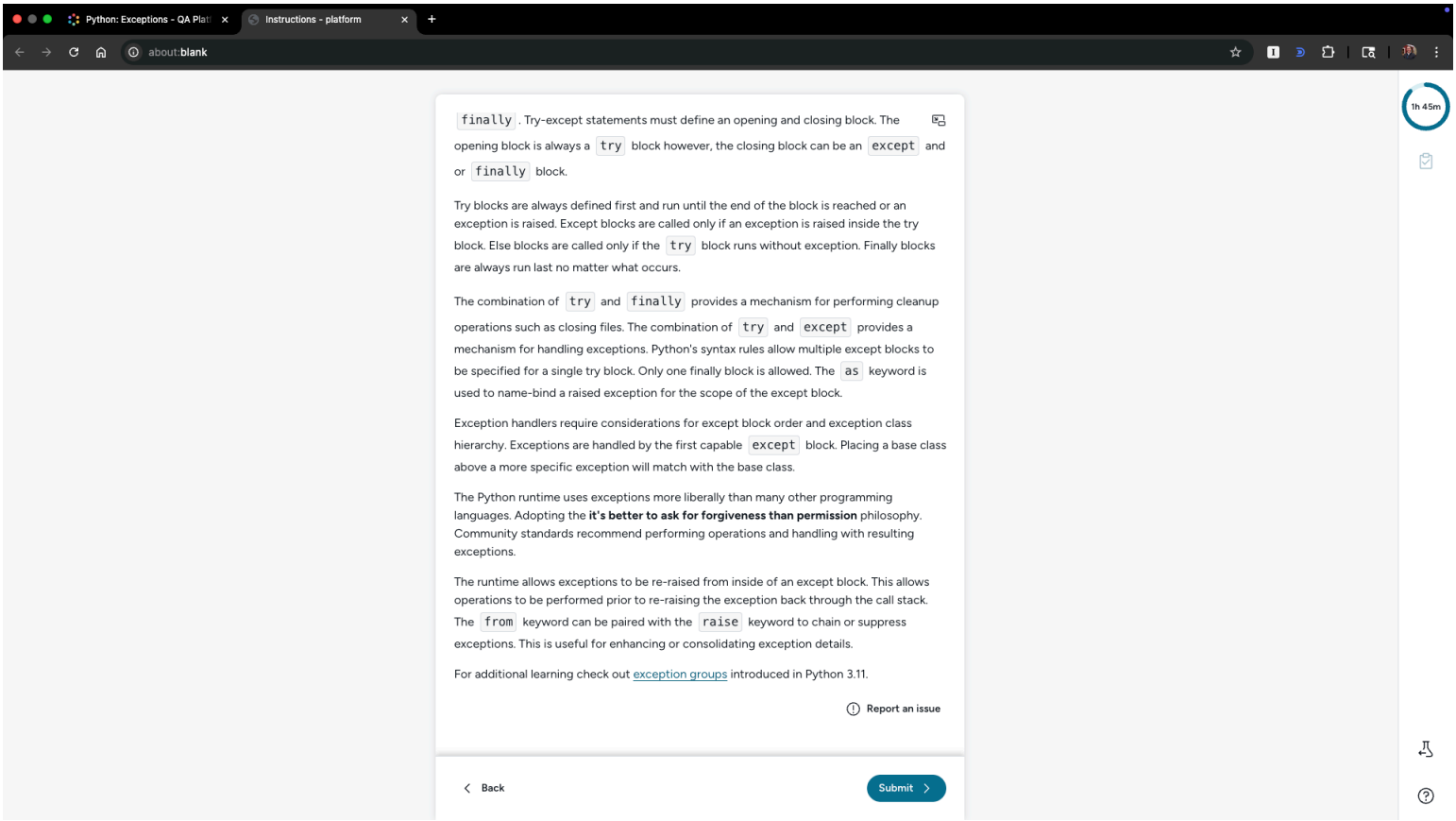

Python: Exceptions - QA Platf
Instructions - platform
about:blank
5 Summary
Introduction
Exceptions disrupt normal code flow in order to report exceptional conditions. Exceptional conditions include failed and interrupted operations. Exceptions inform the Python runtime that the normal code flow cannot continue. The standard library includes a globally available hierarchy of exception types. User-defined exceptions are derived from an existing exception type.
Raising exceptions is accomplished by using the keyword raise followed by either an exception type or instance. Raising an exception by type is used when exceptions require no additional metadata. Raising by instance allows metadata describing the cause of the exception to be captured. Exceptions are normal objects until they're raised. Unhandled exceptions cause the application to exit after writing the exception and traceback to the console.
The try-except statement includes the keywords try , except , else , and finally . Try-except statements must define an opening and closing block. The opening block is always a try block however, the closing block can be an except and or finally block.
Try blocks are always defined first and run until the end of the block is reached or an exception is raised. Except blocks are called only if an exception is raised inside the try block. Else blocks are called only if the try block runs without exception. Finally blocks are always run last no matter what occurs.
The combination of try and finally provides a mechanism for performing cleanup operations such as closing files. The combination of try and except provides a mechanism for handling exceptions. Python's syntax rules allow multiple except blocks to be specified for a single try block. Only one finally block is allowed. The as keyword is used to name-bind a raised exception for the scope of the except block.
Exception handlers require considerations for except block order and exception class
Back
Submit
1h 45m


Python: Exceptions - QA Platf
Instructions - platform
about:blank
finally . Try-except statements must define an opening and closing block. The opening block is always a try block however, the closing block can be an except and or finally block.
Try blocks are always defined first and run until the end of the block is reached or an exception is raised. Except blocks are called only if an exception is raised inside the try block. Else blocks are called only if the try block runs without exception. Finally blocks are always run last no matter what occurs.
The combination of try and finally provides a mechanism for performing cleanup operations such as closing files. The combination of try and except provides a mechanism for handling exceptions. Python's syntax rules allow multiple except blocks to be specified for a single try block. Only one finally block is allowed. The as keyword is used to name-bind a raised exception for the scope of the except block.
Exception handlers require considerations for except block order and exception class hierarchy. Exceptions are handled by the first capable except block. Placing a base class above a more specific exception will match with the base class.
The Python runtime uses exceptions more liberally than many other programming languages. Adopting the it's better to ask for forgiveness than permission philosophy. Community standards recommend performing operations and handling with resulting exceptions.
The runtime allows exceptions to be re-raised from inside of an except block. This allows operations to be performed prior to re-raising the exception back through the call stack. The from keyword can be paired with the raise keyword to chain or suppress exceptions. This is useful for enhancing or consolidating exception details.
For additional learning check out exception groups introduced in Python 3.11.
Report an issue
Back
Submit
1h 45m