```
(base) PS C:\Users\Abi Rahman> python
Python 3.10.9 | packaged by conda-forge | (main, Jan 11 2023, 15:15:40) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> #8. Errors and Exceptions
>>> #8.1. Syntax Errors
>>> while True print('Hello world')
  File "<stdin>", line 1
    while True print('Hello world')
               ^^^^^
SyntaxError: invalid syntax
>>>
>>> #8.2. Exceptions
>>> 10 * (1/0)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ZeroDivisionError: division by zero
>>> 4 + spam*3
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'spam' is not defined
>>> '2' + 2
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: can only concatenate str (not "int") to str
>>>
>>>
>>> #8.3. Handling Exceptions
>>> while True:
...     try:
...         x = int(input("Please enter a number: "))
```

```
...         break
...     except ValueError:
...         print("Oops!  That was no valid number.  Try again...")
...
Please enter a number: 24
>>>
>>> while True:
...     try:
...         x = int(input("Please enter a number: "))
...         break
...     except ValueError:
...         print("Oops!  That was no valid number.  Try again...")
...     except (RuntimeError, TypeError, NameError):
...         pass
...     class B(Exception):
...         pass
...     class C(B):
...         pass
...     class D(C):
...         pass
...
Please enter a number: 24
>>> for cls in [B, C, D]:
...     try:
...         raise cls()
...     except D:
...         print("D")
...     except C:
...         print("C")
...     except B:
...         print("B")
```

```
...
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'B' is not defined
>>>
>>> try:
...     raise Exception('spam', 'eggs')
... except Exception as inst:
...     print(type(inst))    # the exception type
...     print(inst.args)     # arguments stored in .args
...     print(inst)          # __str__ allows args to be printed directly,
...                          # but may be overridden in exception subclasses
...     x, y = inst.args     # unpack args
...     print('x =', x)
...     print('y =', y)
...
<class 'Exception'>
('spam', 'eggs')
('spam', 'eggs')
x = spam
y = eggs
>>>
>>>
>>> import sys
>>> try:
...     f = open('myfile.txt')
...     s = f.readline()
...     i = int(s.strip())
... except OSError as err:
...     print("OS error:", err)
... except ValueError:
```

```
...     print("Could not convert data to an integer.")
... except Exception as err:
...     print(f"Unexpected {err=}, {type(err)=}")
...     raise
...
OS error: [Errno 2] No such file or directory: 'myfile.txt'
>>> for arg in sys.argv[1:]:
...     try:
...         f = open(arg, 'r')
...     except OSError:
...         print('cannot open', arg)
...     else:
...         print(arg, 'has', len(f.readlines()), 'lines')
...         f.close()
...
>>> def this_fails():
...     x = 1/0
...
>>> try:
...     this_fails()
... except ZeroDivisionError as err:
...     print('Handling run-time error:', err)
...
Handling run-time error: division by zero
>>>
>>> #8.4. Raising Exceptions
>>> raise NameError('HiThere')
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: HiThere
>>> raise ValueError  # shorthand for 'raise ValueError()'
```

```
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError
>>>
>>> try:
...     raise NameError('HiThere')
... except NameError:
...     print('An exception flew by!')
...     raise
...
An exception flew by!
Traceback (most recent call last):
  File "<stdin>", line 2, in <module>
NameError: HiThere
>>>
>>>
>>> #8.5. Exception Chaining
>>> try:
...     open("database.sqlite")
... except OSError:
...     raise RuntimeError("unable to handle error")
...
Traceback (most recent call last):
  File "<stdin>", line 2, in <module>
FileNotFoundError: [Errno 2] No such file or directory: 'database.sqlite'

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "<stdin>", line 4, in <module>
RuntimeError: unable to handle error
```

```
>>>
>>> # exc must be exception instance or None.
>>> raise RuntimeError from exc
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'exc' is not defined. Did you mean: 'exec'?
>>> def func():
...     raise ConnectionError
...
>>> try:
...     func()
... except ConnectionError as exc:
...     raise RuntimeError('Failed to open database') from exc
...
Traceback (most recent call last):
  File "<stdin>", line 2, in <module>
  File "<stdin>", line 2, in func
ConnectionError

The above exception was the direct cause of the following exception:

Traceback (most recent call last):
  File "<stdin>", line 4, in <module>
RuntimeError: Failed to open database
>>>
>>> try:
...     open('database.sqlite')
... except OSError:
...     raise RuntimeError from None
...
Traceback (most recent call last):
```

```
  File "<stdin>", line 4, in <module>
RuntimeError
>>>
>>> #8.7. Defining Clean-up Actions
>>> try:
...     raise KeyboardInterrupt
... finally:
...     print('Goodbye, world!')
...
Goodbye, world!
Traceback (most recent call last):
  File "<stdin>", line 2, in <module>
KeyboardInterrupt
>>>
>>> def bool_return():
...     try:
...         return True
...     finally:
...         return False
...
>>> bool_return()
False
>>>
>>> def divide(x, y):
...     try:
...         result = x / y
...     except ZeroDivisionError:
...         print("division by zero!")
...     else:
...         print("result is", result)
...     finally:
```

```
...         print("executing finally clause")
...
>>> divide(2, 1)
result is 2.0
executing finally clause
>>> divide(2, 0)
division by zero!
executing finally clause
>>> divide("2", "1")
executing finally clause
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "<stdin>", line 3, in divide
TypeError: unsupported operand type(s) for /: 'str' and 'str'
>>>
>>> #8.8. Predefined Clean-up Actions
>>> for line in open("myfile.txt"):
...     print(line, end="")
...
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
FileNotFoundError: [Errno 2] No such file or directory: 'myfile.txt'
>>>
>>> with open("myfile.txt") as f:
...     for line in f:
...         print(line, end="")
...
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
FileNotFoundError: [Errno 2] No such file or directory: 'myfile.txt'
>>>
```

```
>>> #8.9. Raising and Handling Multiple Unrelated Exceptions
>>> def f():
...     excs = [OSError('error 1'), SystemError('error 2')]
...     raise ExceptionGroup('there were problems', excs)
...
>>> f()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "<stdin>", line 3, in f
NameError: name 'ExceptionGroup' is not defined
>>> try:
...     f()
... except Exception as e:
...     print(f'caught {type(e)}: e')
...
caught <class 'NameError'>: e
>>>
>>> def f():
...     raise ExceptionGroup(
...         "group1",
...         [
...             OSError(1),
...             SystemError(2),
...             ExceptionGroup(
...                 "group2",
...                 [
...                     OSError(3),
...                     RecursionError(4)
...                 ]
...             )
...         ]
```

```
...     )
...
>>> excs = []
>>> for test in tests:
...     try:
...         test.run()
...     except Exception as e:
...         excs.append(e)
...
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'tests' is not defined
>>> if excs:
...    raise ExceptionGroup("Test Failures", excs)
...
>>>
>>>
>>> #8.10. Enriching Exceptions with Notes
>>> try:
...     raise TypeError('bad type')
... except Exception as e:
...     e.add_note('Add some information')
...     e.add_note('Add some more information')
...     raise
...
Traceback (most recent call last):
  File "<stdin>", line 2, in <module>
TypeError: bad type


During handling of the above exception, another exception occurred:
```

```
Traceback (most recent call last):
  File "<stdin>", line 4, in <module>
AttributeError: 'TypeError' object has no attribute 'add_note'
>>>
>>> def f():
...     raise OSError('operation failed')
...
>>>
>>> excs = []
>>> for i in range(3):
...     try:
...         f()
...     except Exception as e:
...         e.add_note(f'Happened in Iteration {i+1}')
...         excs.append(e)
...
Traceback (most recent call last):
  File "<stdin>", line 3, in <module>
  File "<stdin>", line 2, in f
OSError: operation failed

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "<stdin>", line 5, in <module>
AttributeError: 'OSError' object has no attribute 'add_note'
>>> raise ExceptionGroup('We have some problems', excs)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'ExceptionGroup' is not defined
>>>
```