ComIT

Execution Flow II – Error Handling

Errors and Exceptions

- There are times during the life of your program where it will enter a state where it can no longer execute the code as intended.
- In these situations, the Python runtime will generate an error and terminate the program.
- We call these errors “Exceptions”, and they alter the normal program flow.
- In order to make our programs stable (not crash), we need to handle these exceptions and recover from them when possible.

Exception Handling

- Exceptions can be handled by using the **try** keyword in combination with the **except** keyword.

```
try:
    print(“Hello World!”)
except as e:
    print(“An error occurred: ”, e)
```

- The code in the “try” block is attempted, and if an error occurs the code in the try block will stop and program execution flow will resume in the “except” block.

Exception Handling

- An **else** block can be added, and it will only execute when no error was produced:

```
try:
    print(“Hello World!”)
except as e:
    print(“An error occurred: ”, e)
else:
    print(“No error here!”)
```

Exception Handling

- A **finally** block can also be added, and it will execute regardless of whether or not an error was produced:

```
try:
    print(“Hello World!”)
except as e:
    print(“An error occurred: ”, e)
finally:
    print(“Can’t stop this code!!”)
```

The Call Stack

- Before we learn more about exceptions, we need a little more understanding of how Python keeps track of what function is currently running.
- It uses a “first in, first out” (FIFO) data structure known as a “call stack”.
- It is a simple record of all functions that are currently being called.
- Every time a function execution starts, the record is “pushed” onto the stack.
- When a function returns, it is popped off the stack.

The Call Stack - Errors

- When an exception is raised, the Python interpreter starts to “tear down” the program until it is inside the scope of a try-except block that is capable of handling the error.
- This “tearing down” is actually just popping each item off the top of the call stack.
- If every function is removed from the stack, what happens?

Error Types

- There are actually many different types of errors built into Python, and you can even define your own.
- A try-except block will only handle errors that match the type you are trying to handle.
- You can define many **except** blocks for a single **try** block, and you can resume program flow differently based on what error was thrown.
- Sometimes it’s best not to handle every error type and to let the exception propagate through the stack!

Questions?