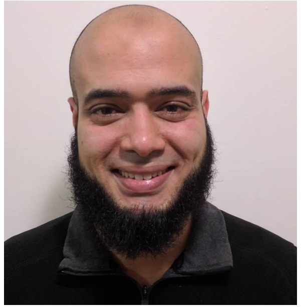
Python Programming

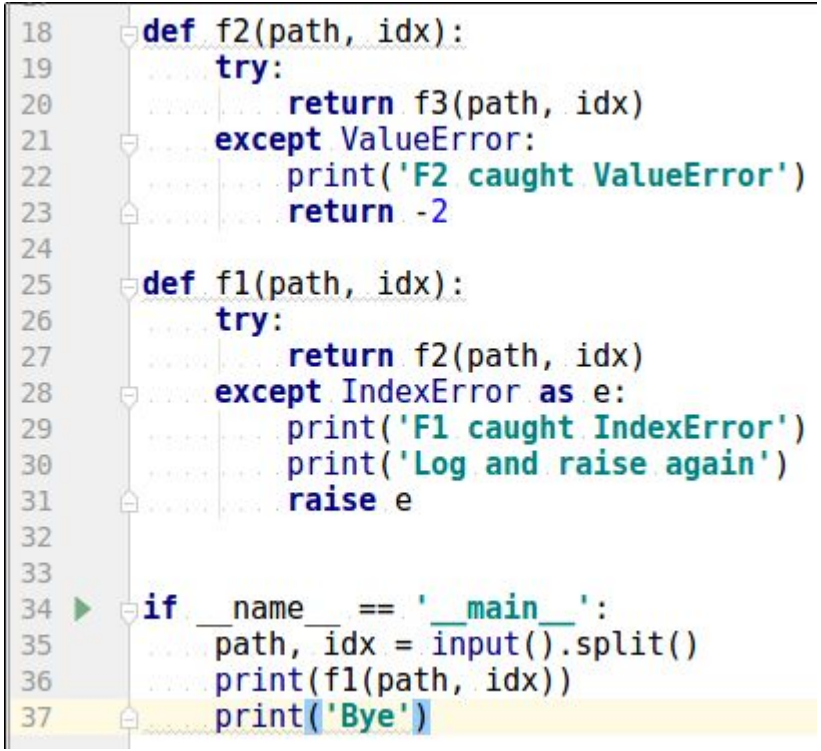
Stack Unwinding

Mostafa S. Ibrahim
Teaching, Training and Coaching since more than a decade!

Artificial Intelligence & Computer Vision Researcher
PhD from Simon Fraser University - Canada
Bachelor / Msc from Cairo University - Egypt
Ex-(Software Engineer / ICPC World Finalist)



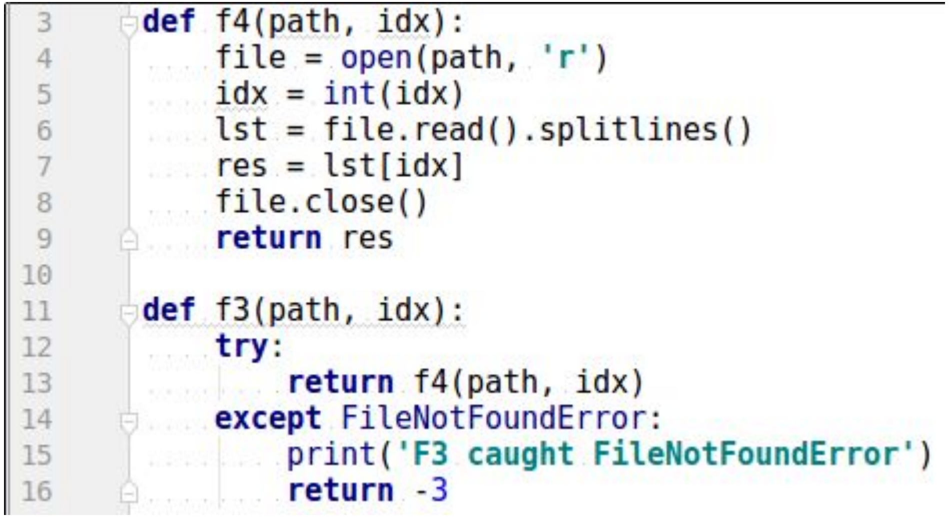
Who catch the exception?



```python
def f2(path, idx):
    try:
        return f3(path, idx)
    except ValueError:
        print('F2 caught ValueError')
        return -2

def f1(path, idx):
    try:
        return f2(path, idx)
    except IndexError as e:
        print('F1 caught IndexError')
        print('Log and raise again')
        raise e


if __name__ == '__main__':
    path, idx = input().split()
    print(f1(path, idx))
    print('Bye')
```



```python
def f4(path, idx):
    file = open(path, 'r')
    idx = int(idx)
    lst = file.read().splitlines()
    res = lst[idx]
    file.close()
    return res

def f3(path, idx):
    try:
        return f4(path, idx)
    except FileNotFoundError:
        print('F3 caught FileNotFoundError')
        return -3
```

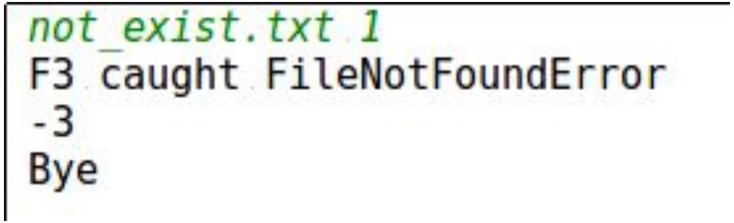
Stack Unwinding

- Assume some function1 calls function2:
- Assume function2 throws an exception:
    - Does it have a try catch surrounding its location?
        - No ⇒ Terminate the function, remove function2 from stack
        - Yes ⇒ Does it have a matching type?
            - No ⇒ Same as last NO
            - Yes ⇒ Go to its except block (which may throw again)
- If function1 receives a thrown exception from function 1
    - It does the same logic
    - Either catch and end it or terminate function and propagate up to the caller
- If caller script received a thrown exception and did not handle:
    - program crashes at this point

Trace: not_exist.txt 1

- F4: May throw several errors
- F3: Catch **File not found Error**
- F2: Catch Value Error
- F1: Catch Index Error / Rethrow
- Main: No global catch

- F4 throws FileNotFoundError
- F3 catches it and return -3
- Program exists without exceptions

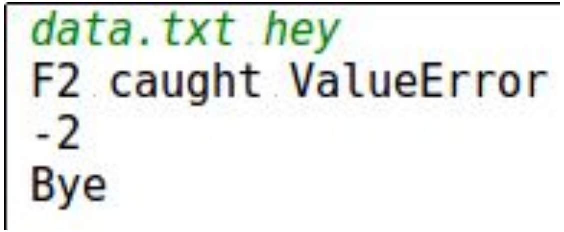


```
not_exist.txt 1
F3 caught FileNotFoundError
-3
Bye
```

Trace: data.txt hey

- F4: May throw several errors
- F3: Catch File not found Error
- F2: Catch **Value Error**
- F1: Catch Index Error / Rethrow
- Main: No global catch

- F4 throws ValueError
- F3 can’t catch it
    - It propagates up to F2
- F2 catches it
- Program exists without exceptions



```
data.txt hey
F2 caught ValueError
-2
Bye
```

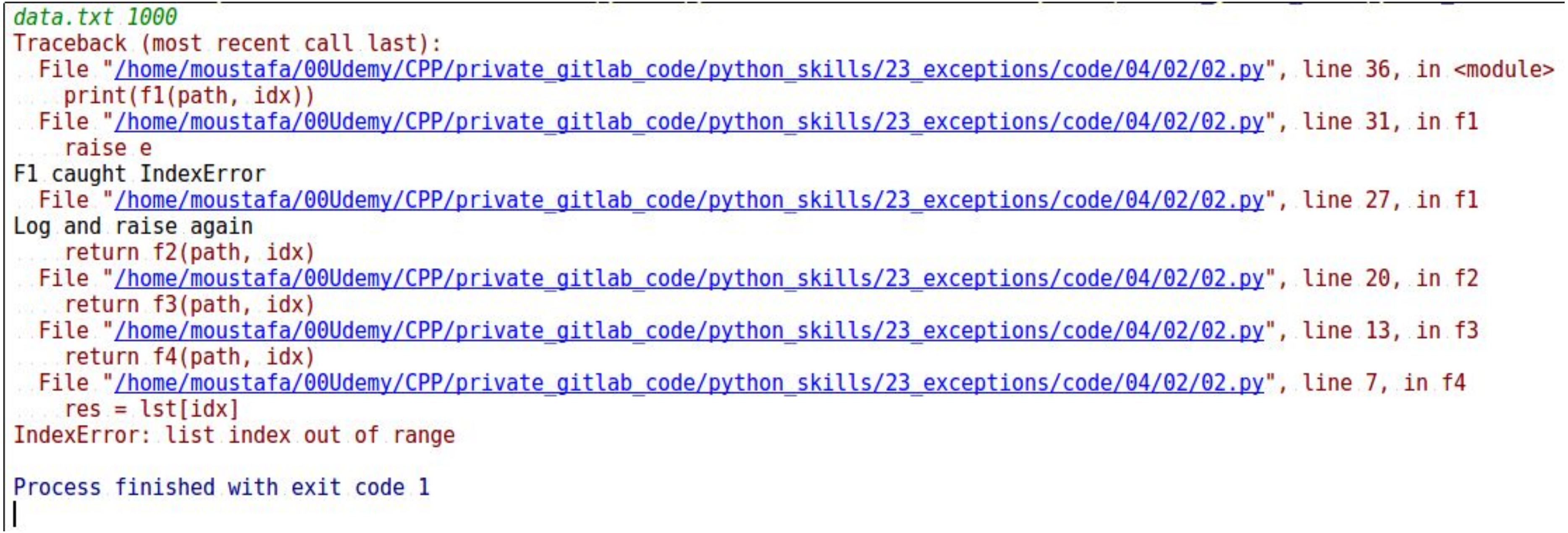
Trace: data.txt 1000

- F4: May throw several errors
- F3: Catch File not found Error
- F2: Catch Value Error
- F1: Catch **Index Error** / Rethrow
- Main: No global catch

- F4 throws IndexError
- F3 can’t catch it: Propagate up
- F2 can’t catch it: Propagate up
- F1 catches it, but raise again
- Program exists an exception

Trace: data.txt 1000

- Observe: Exit code 1.
- Observe the traceback



```
data.txt 1000
Traceback (most recent call last):
  File "/home/moustafa/00Udemy/CPP/private_gitlab_code/python_skills/23_exceptions/code/04/02/02.py", line 36, in <module>
    print(f1(path, idx))
  File "/home/moustafa/00Udemy/CPP/private_gitlab_code/python_skills/23_exceptions/code/04/02/02.py", line 31, in f1
    raise e
F1 caught IndexError
  File "/home/moustafa/00Udemy/CPP/private_gitlab_code/python_skills/23_exceptions/code/04/02/02.py", line 27, in f1
Log and raise again
    return f2(path, idx)
  File "/home/moustafa/00Udemy/CPP/private_gitlab_code/python_skills/23_exceptions/code/04/02/02.py", line 20, in f2
    return f3(path, idx)
  File "/home/moustafa/00Udemy/CPP/private_gitlab_code/python_skills/23_exceptions/code/04/02/02.py", line 13, in f3
    return f4(path, idx)
  File "/home/moustafa/00Udemy/CPP/private_gitlab_code/python_skills/23_exceptions/code/04/02/02.py", line 7, in f4
    res = lst[idx]
IndexError: list index out of range

Process finished with exit code 1
```

Tips

- Don’t throw an exception for a normal flow
  - Exception means something blocking!
- Swallowing Exceptions
  - Don’t hide it.
  - Either don’t catch
  - Or catch
    - Handle it
    - Or log and Raise
- Be careful from code inside except/else/finally to raise another exception unintentionally

"Acquire knowledge and impart it to the people."

"Seek knowledge from the Cradle to the Grave."