Python fundamentals day 62

Today's agenda

- Exception handling contd

Exception handling contd

Let us take another example

```
print('Execution started normally')
lst=[10,20,0,40,50]
d={1:'c',2:'java',3:'python',4:'c++'}

try:
    r=int(input('Enter the rank of language:\n'))
    print(d[r])
    num=int(input('Enter the index of numerator:'))
    den=int(input('Enter the index of denominator:'))
    print(lst[num]/lst[den])
except:
    print('Hey there was an issue!')

print('Execution terminated normally')
```

Output:

```
In [1]: runfile('C:/Users/rooman/OneDrive/Desktop/python/
test1.py', wdir='C:/Users/rooman/OneDrive/Desktop/python')
Execution started normally

Enter the rank of language:
3
python

Enter the index of numerator:3

Enter the index of denominator:0
4.0
Execution terminated normally
```

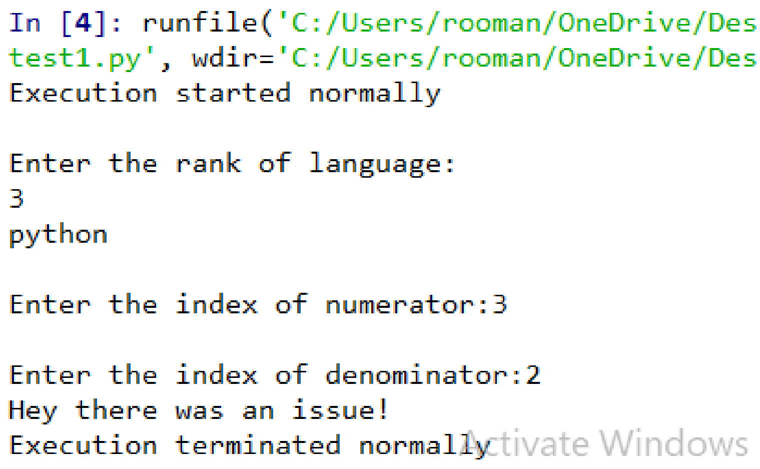
This is the ideal case. Let us try with other inputs

```
In [2]: runfile('C:/Users/rooman/OneDrive/Desktop/python/
test1.py', wdir='C:/Users/rooman/OneDrive/Desktop/python')
Execution started normally

Enter the rank of language:
5
Hey there was an issue!
Execution terminated normally
```

In the above case we have given the key which is not present in out dictionary. Therefore keyerror exception object is generated and it checks for except block and completes execution. But the message is not very clear. Let us see another case

```
In [3]: runfile('C:/Users/rooman/OneDrive/Desktop/python/
test1.py', wdir='C:/Users/rooman/OneDrive/Desktop/python')
Execution started normally

Enter the rank of language:
3
python

Enter the index of numerator:5

Enter the index of denominator:3
Hey there was an issue!
Execution terminated normally
```

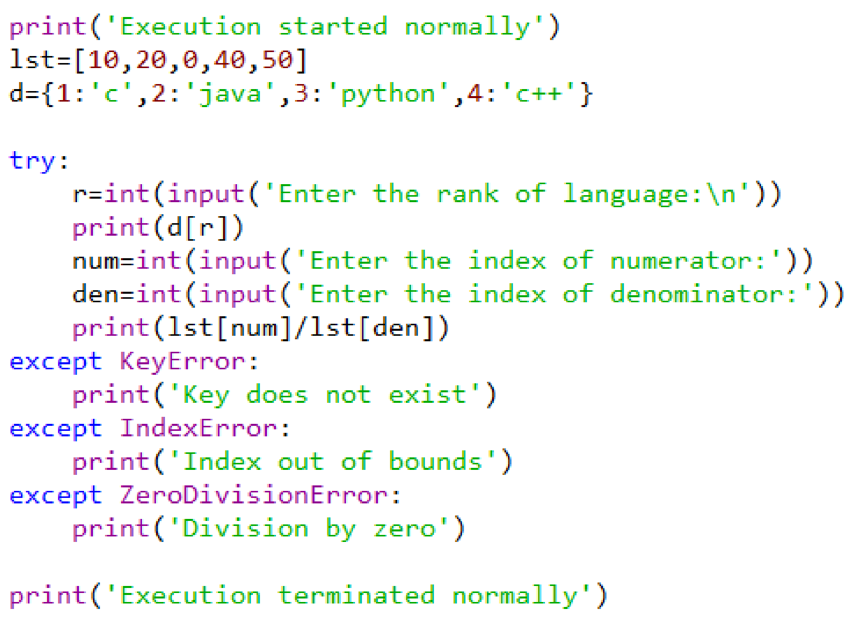
In the above case we don't have 5th index in list. Therefore while performing division indexerror exception object is generated and it checks for the except block and completes the execution. But here as well we get the same message and it doesn't convey the clear message. Let us check for another case



```
In [4]: runfile('C:/Users/rooman/OneDrive/Desktop/python/
test1.py', wdir='C:/Users/rooman/OneDrive/Desktop/python')
Execution started normally

Enter the rank of language:
3
python

Enter the index of numerator:3

Enter the index of denominator:2
Hey there was an issue!
Execution terminated normally
```

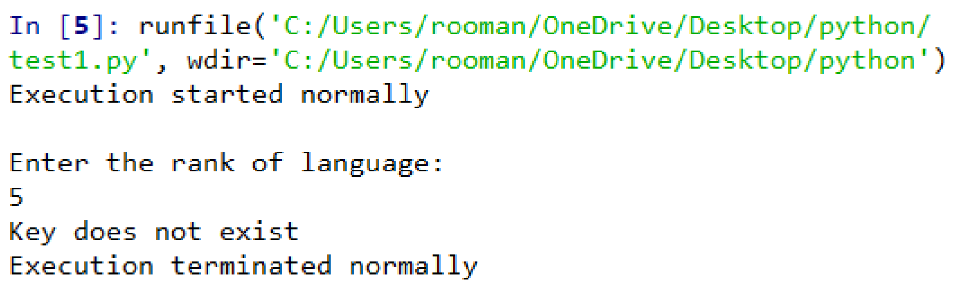
Here we are trying to divide 3rd index by 2nd. 2nd index is 0 which will result in zerodivision error exception which will check for except block and return the message.

But as seen in above cases all exceptions generated the same message which is not conveying the issue properly and hence we need multiple except block for respective exceptions. As shown below



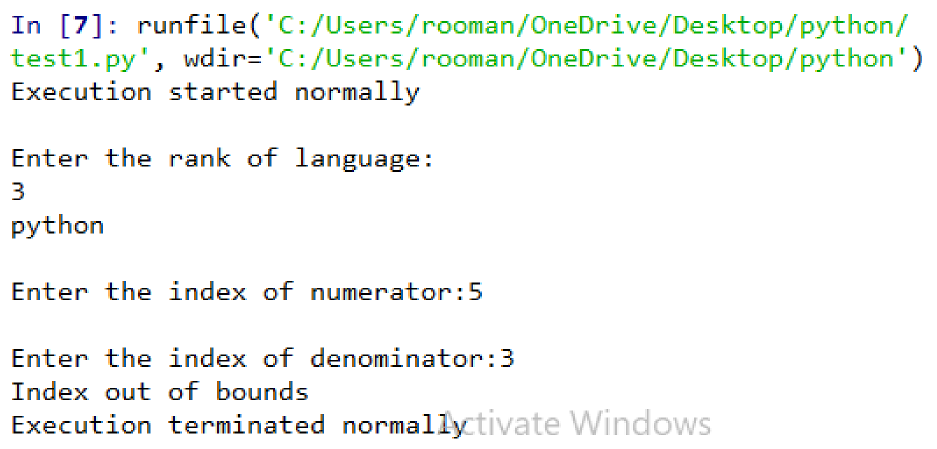
```
print('Execution started normally')
lst=[10,20,0,40,50]
d={1:'c',2:'java',3:'python',4:'c++'}

try:
    r=int(input('Enter the rank of language:\n'))
    print(d[r])
    num=int(input('Enter the index of numerator:'))
    den=int(input('Enter the index of denominator:'))
    print(lst[num]/lst[den])
except KeyError:
    print('Key does not exist')
except IndexError:
    print('Index out of bounds')
except ZeroDivisionError:
    print('Division by zero')

print('Execution terminated normally')
```

Let us try giving the same previous inputs and check the output now



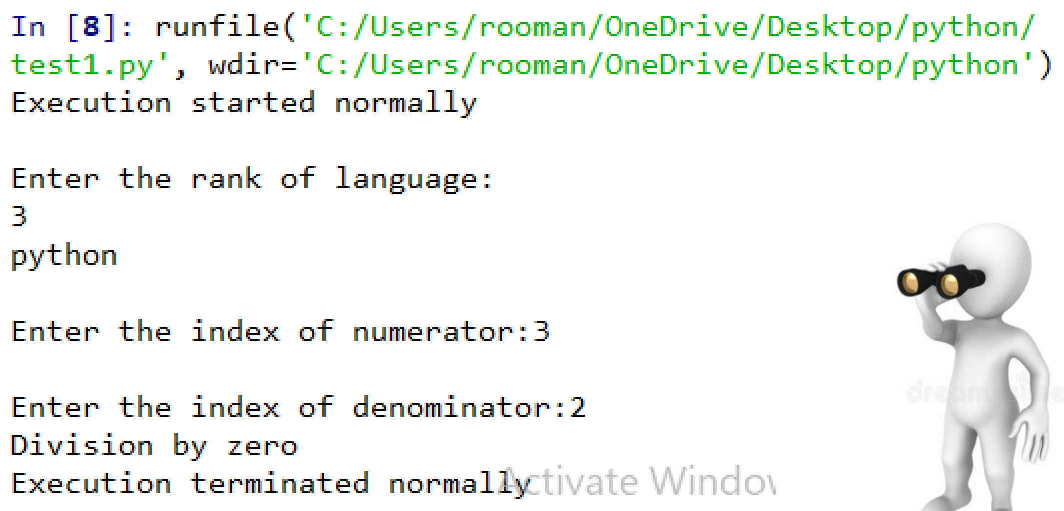
```
In [5]: runfile('C:/Users/rooman/OneDrive/Desktop/python/
test1.py', wdir='C:/Users/rooman/OneDrive/Desktop/python')
Execution started normally

Enter the rank of language:
5
Key does not exist
Execution terminated normally
```

Great! Now we know the entered key does not exist. And can correct it by giving proper key. And note that once a particular except block is executed other except blocks do not execute.



```
In [7]: runfile('C:/Users/rooman/OneDrive/Desktop/python/
test1.py', wdir='C:/Users/rooman/OneDrive/Desktop/python')
Execution started normally

Enter the rank of language:
3
python

Enter the index of numerator:5

Enter the index of denominator:3
Index out of bounds
Execution terminated normally
```

Now we know the issue is with the index number. We can now give appropriate input.



```
In [8]: runfile('C:/Users/rooman/OneDrive/Desktop/python/
test1.py', wdir='C:/Users/rooman/OneDrive/Desktop/python')
Execution started normally

Enter the rank of language:
3
python

Enter the index of numerator:3

Enter the index of denominator:2
Division by zero
Execution terminated normally
```

And now the issue is with Zero division.

But the above except block will catch only specific exception objects. What if an exception generated cannot be handled by any of the above except blocks? Let us check

```
  File "C:\Users\rooman\Anaconda3\lib\site-packages
\spyder_kernels\customize\spydercustomize.py", line 110,
in execfile
    exec(compile(f.read(), filename, 'exec'), namespace)

  File "C:/Users/rooman/OneDrive/Desktop/python/test1.py",
line 6, in <module>
    r=int(input('Enter the rank of language:\n'))

ValueError: invalid literal for int() with base 10: 'five'
```

Certainly abrupt termination occurs. To avoid this it is a good habit to add a generic except block which accept all the possible exception objects as shown below

```
print('Execution started normally')
lst=[10,20,0,40,50]
d={1:'c',2:'java',3:'python',4:'c++'}

try:
    r=int(input('Enter the rank of language:\n'))
    print(d[r])
    num=int(input('Enter the index of numerator:'))
    den=int(input('Enter the index of denominator:'))
    print(lst[num]/lst[den])
except KeyError:   #specific handlers
    print('Key does not exist')
except IndexError:
    print('Index out of bounds')
except ZeroDivisionError:
    print('Division by zero')
except:   #generic handler
    print('Hey some issue occurred')

print('Execution terminated normally')
```

```
In [10]: runfile('C:/Users/rooman/OneDrive/Desktop/python/
test1.py', wdir='C:/Users/rooman/OneDrive/Desktop/python')
Execution started normally

Enter the rank of language:
five
Hey some issue occurred
Execution terminated normally
```

Great! Now we know how to tackle all problems. But now what if you want to place the generic except bock on top of all except blocks;

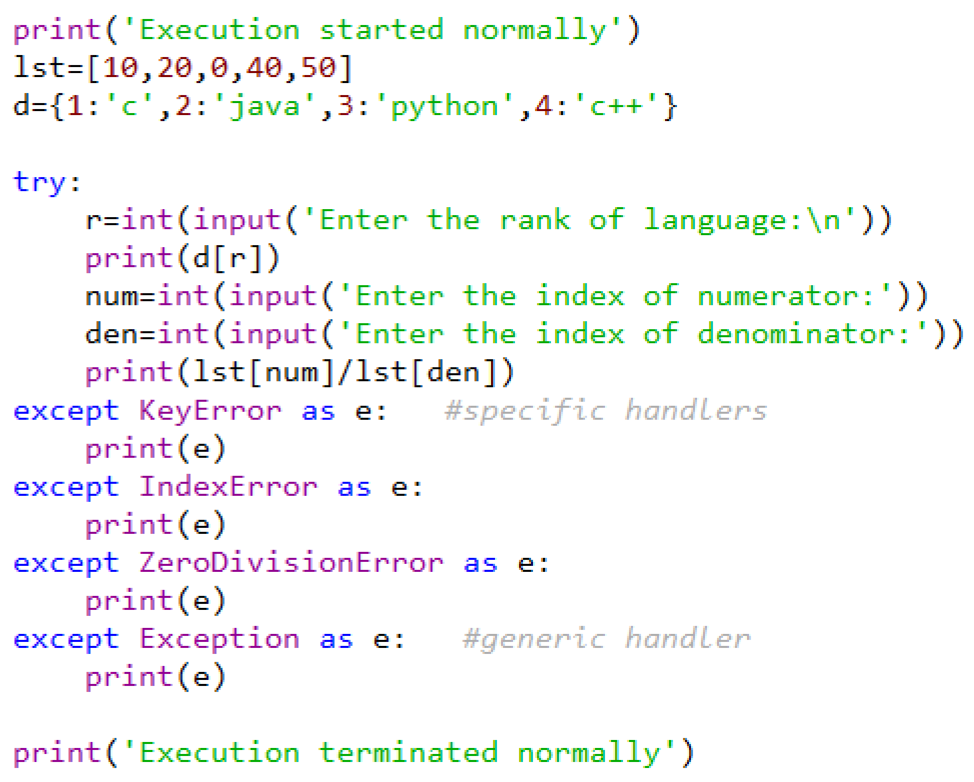
```
print('Execution started normally')
lst=[10,20,0,40,50]
d={1:'c',2:'java',3:'python',4:'c++'}

try:
    r=int(input('Enter the rank of language:\n'))
    print(d[r])
    num=int(input('Enter the index of numerator:'))
    den=int(input('Enter the index of denominator:'))
    print(lst[num]/lst[den])
except:    #generic handler
    print('Hey some issue occurred')
except KeyError:    #specific handlers
    print('Key does not exist')
except IndexError:
    print('Index out of bounds')
except ZeroDivisionError:
    print('Division by zero')


print('Execution terminated normally')
```

Output:

```
  File "C:\Users\rooman\Anaconda3\lib\site-packages
\spyder_kernels\customize\spydercustomize.py", line 110,
in execfile
    exec(compile(f.read(), filename, 'exec'), namespace)

  File "C:/Users/rooman/OneDrive/Desktop/python/test1.py",
line 10
    print(lst[num]/lst[den])
                           ^
SyntaxError: default 'except:' must be last
```

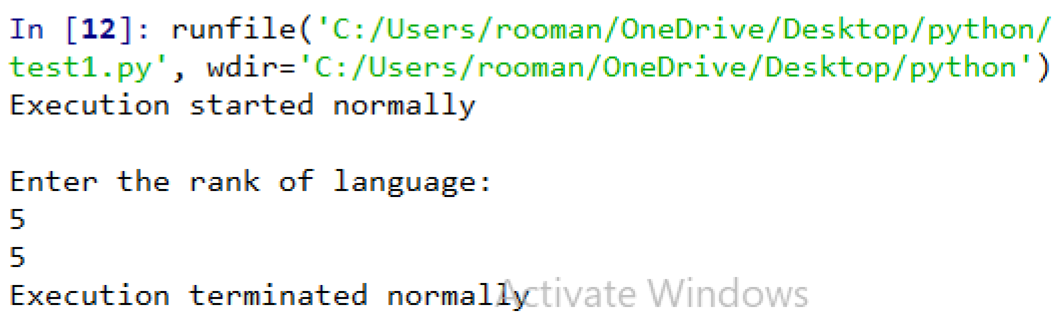
Well that will give you an error, because when an exception object is generated it checks for except blocks sequentially therefore once the generic except block catches the exception, other specific handlers will never get chance to execute, therefore generic handlers are suppose be at the end

Next let us try to access the exception object and print its default message



```
print('Execution started normally')
lst=[10,20,0,40,50]
d={1:'c',2:'java',3:'python',4:'c++'}

try:
    r=int(input('Enter the rank of language:\n'))
    print(d[r])
    num=int(input('Enter the index of numerator:'))
    den=int(input('Enter the index of denominator:'))
    print(lst[num]/lst[den])
except KeyError as e:   #specific handlers
    print(e)
except IndexError as e:
    print(e)
except ZeroDivisionError as e:
    print(e)
except Exception as e:   #generic handler
    print(e)

print('Execution terminated normally')
```

Let us give the same inputs and check the message.



```
In [12]: runfile('C:/Users/rooman/OneDrive/Desktop/python/
test1.py', wdir='C:/Users/rooman/OneDrive/Desktop/python')
Execution started normally

Enter the rank of language:
5
5
Execution terminated normally
```

It just displays the key when keyerror is encountered.

```
In [13]: runfile('C:/Users/rooman/OneDrive/Desktop/python/
test1.py', wdir='C:/Users/rooman/OneDrive/Desktop/python')
Execution started normally

Enter the rank of language:
3
python

Enter the index of numerator:5

Enter the index of denominator:3
list index out of range
Execution terminated normally
```

Okay this is an appropriate message. Great!

```
In [14]: runfile('C:/Users/rooman/OneDrive/Desktop/python/
test1.py', wdir='C:/Users/rooman/OneDrive/Desktop/python')
Execution started normally

Enter the rank of language:
3
python

Enter the index of numerator:3

Enter the index of denominator:2
division by zero
Execution terminated normally
```

Appropriate message here as well

```
In [15]: runfile('C:/Users/rooman/OneDrive/Desktop/python/
test1.py', wdir='C:/Users/rooman/OneDrive/Desktop/python')
Execution started normally

Enter the rank of language:
five
invalid literal for int() with base 10: 'five'
Execution terminated normally
```

So we are getting appropriate messages. So it is up to the user which one they want to go with.