27.ExceptionHandling

June 11, 2020

OOPS

Bank Application using OOPs

0.0.1 Exceptions Handling

[1]:
```
name
```

```
␣
↪---------------------------------------------------------------------------

NameError                                 Traceback (most recent call␣
↪last)

<ipython-input-1-9bc0cb2ed6de> in <module>
----> 1 name


NameError: name 'name' is not defined
```

[2]:
```
1 / 0
```

```
␣
↪---------------------------------------------------------------------------

ZeroDivisionError                         Traceback (most recent call␣
↪last)

<ipython-input-2-bc757c3fda29> in <module>
----> 1 1 / 0


ZeroDivisionError: division by zero
```

[3]:
```
x = int(input())
```

```
lskdfj
```

```
␣
↪---------------------------------------------------------------------------

ValueError                                Traceback (most recent call␣
↪last)

<ipython-input-3-b18148c418c0> in <module>
----> 1 x = int(input())


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

[7]:
```
if True:
 print('hello world')
    print("hi")
```

```
  File "<ipython-input-7-98e23201d5d4>", line 3
    print("hi")
    ^
IndentationError: unexpected indent
```

[ ]:
```
fp = open('alkdfj')
```

Exceptions are run time errors caused by some logical reason

[10]:
```
x = int(input("Enter x: ")) # value error
y = int(input("Enter y: ")) # value error
result = x / y # zero division error

print("X: ", x)
print("Y: ", y)
print("Result: ", result)
```

```
Enter x: alskdfj
```

```
␣
↪---------------------------------------------------------------------------
```

```
        ValueError                                Traceback (most recent call␣
↪last)

        <ipython-input-10-84a58e9933e1> in <module>
   ----> 1 x = int(input("Enter x: ")) # value error
         2 y = int(input("Enter y: ")) # value error
         3 result = x / y # zero division error
         4
         5 print("X: ", x)


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

[13]:
```
print("Begining".center(60, '_'))
x = int(input("Enter x: ")) # value error
y = int(input("Enter y: ")) # value error
result = x / y # zero division error

print("X: ", x)
print("Y: ", y)
print("Result: ", result)
print("Ending".center(60,'_'))
```

```
___________________________Begning____________________________
Enter x: 1
Enter y: 0


      ␣
↪--------------------------------------------------------------------------

        ZeroDivisionError                         Traceback (most recent call␣
↪last)

        <ipython-input-13-023665a1856e> in <module>
          2 x = int(input("Enter x: ")) # value error
          3 y = int(input("Enter y: ")) # value error
   ----> 4 result = x / y # zero division error
          5
          6 print("X: ", x)


        ZeroDivisionError: division by zero
```

[15]:
```
print("Began".center(60,'_'))
try:
```

```
    x = int(input("Enter x: ")) # value error
    y = int(input("Enter y: ")) # value error
    result = x / y # zero division error

    print("X: ", x)
    print("Y: ", y)
    print("Result: ", result)
except:
    print("!!Error!! Something Went Wrong")
print("End".center(60, '_'))
```

```
___________________________Began____________________________
Enter x: dkfj
!!Error!! Something Went Wrong
____________________________End____________________________
```

[16]:
```
print("Began".center(60,'_'))
try:
    x = int(input("Enter x: ")) # value error
    y = int(input("Enter y: ")) # value error
    result = x / y # zero division error

    print("X: ", x)
    print("Y: ", y)
    print("Result: ", result)
except Exception as e:
    print(f"!!Error!! {e}")
print("End".center(60, '_'))
```

```
___________________________Began____________________________
Enter x: 1
Enter y: 0
!!Error!! division by zero
____________________________End____________________________
```

[17]:
```
print("Began".center(60,'_'))
try:
    x = int(input("Enter x: ")) # value error
    y = int(input("Enter y: ")) # value error
    result = x / y # zero division error

    print("X: ", x)
    print("Y: ", y)
    print("Result: ", result)
except Exception as e:
    print(f"!!Error!! {e}")
print("End".center(60, '_'))
```

```
____________________________Began____________________________
Enter x: lskdfj
!!Error!! invalid literal for int() with base 10: 'lskdfj'
____________________________End____________________________
```

[18]:
```
int('alskdfj')
```

```
␣
↪---------------------------------------------------------------------------

ValueError                                Traceback (most recent call␣
↪last)

<ipython-input-18-4746fc8d3f2b> in <module>
----> 1 int('alskdfj')


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

[19]:
```
1/0
```

```
␣
↪---------------------------------------------------------------------------

ZeroDivisionError                         Traceback (most recent call␣
↪last)

<ipython-input-19-9e1622b385b6> in <module>
----> 1 1/0


ZeroDivisionError: division by zero
```

[21]:
```
print("Began".center(60,'_'))
while True:
    try:
        x = int(input("Enter x: ")) # value error
        y = int(input("Enter y: ")) # value error
        result = x / y # zero division error

        print("X: ", x)
        print("Y: ", y)
        print("Result: ", result)
```

```
        break
    except ValueError as e:
        print(f"!!Error!! Please Give Proper Integer Number as Input...Thanks")
    except ZeroDivisionError as e:
        print(f"!!Error!!Konsa Gola se aaye ho!!zero se koi divide karta hai␣
↪bhala")
    except Exception as e:
        print("!!Error!!You Are Doing Somethind Nesty!!Something Went Wrong")
        print("Show this msg to our mantinance team: ", e)
    print("End".center(60, '_'))
```

```
____________________________Began____________________________
Enter x: aldj
!!Error!! Please Give Proper Integer Number as Input…Thanks
_____________________________End_____________________________
Enter x: 5
Enter y: 0
!!Error!!Konsa Gola aaye ho!!zero se koi divide karta hai bhala
_____________________________End_____________________________
Enter x: 5
Enter y: 6
X:  5
Y:  6
Result:  0.8333333333333334
```

[24]:
```
import builtins
```

[26]:
```
print(*[ err for err in dir(builtins) if err[0].isupper() ], sep='\n')
```

```
ArithmeticError
AssertionError
AttributeError
BaseException
BlockingIOError
BrokenPipeError
BufferError
BytesWarning
ChildProcessError
ConnectionAbortedError
ConnectionError
ConnectionRefusedError
ConnectionResetError
DeprecationWarning
EOFError
Ellipsis
EnvironmentError
Exception
```

```
False
FileExistsError
FileNotFoundError
FloatingPointError
FutureWarning
GeneratorExit
IOError
ImportError
ImportWarning
IndentationError
IndexError
InterruptedError
IsADirectoryError
KeyError
KeyboardInterrupt
LookupError
MemoryError
ModuleNotFoundError
NameError
None
NotADirectoryError
NotImplemented
NotImplementedError
OSError
OverflowError
PendingDeprecationWarning
PermissionError
ProcessLookupError
RecursionError
ReferenceError
ResourceWarning
RuntimeError
RuntimeWarning
StopAsyncIteration
StopIteration
SyntaxError
SyntaxWarning
SystemError
SystemExit
TabError
TimeoutError
True
TypeError
UnboundLocalError
UnicodeDecodeError
UnicodeEncodeError
UnicodeError
UnicodeTranslateError
```

```
UnicodeWarning
UserWarning
ValueError
Warning
WindowsError
ZeroDivisionError
```

[29]:
```python
print("Began".center(60,'_'))
try:
    x = int(input("Enter x: ")) # value error
    y = int(input("Enter y: ")) # value error
    result = x / y # zero division error
except ZeroDivisionError as e:
    print(f"!!Error!! Invalid Operation can not divide by zero!!")
except ValueError as e:
    print(f"!!Error!! Invalid Input, Input should be a number!!")
except Exception as e:
    print(f"!!Error!! Something Went Wrong {e}")
else:
    print("X: ", x)
    print("Y: ", y)
    print("Result: ", result)
print("End".center(60, '_'))
```

```
____________________________Began____________________________
Enter x: 1
Enter y: 6
X:  1
Y:  6
Result:  0.16666666666666666
_____________________________End_____________________________
```

[30]:
```python
print("Began".center(60,'_'))
try:
    x = int(input("Enter x: ")) # value error
    y = int(input("Enter y: ")) # value error
    result = x / y # zero division error
except ZeroDivisionError as e:
    print(f"!!Error!! Invalid Operation can not divide by zero!!")
except ValueError as e:
    print(f"!!Error!! Invalid Input, Input should be a number!!")
except Exception as e:
    print(f"!!Error!! Something Went Wrong {e}")
else:
    print("X: ", x)
    print("Y: ", y)
    print("Result: ", result)
```

```
print("End".center(60, '_'))
```

```
____________________________Began____________________________
Enter x: 1
Enter y: 0
!!Error!! Invalid Operation can not divide by zero!!
_____________________________End_____________________________
```

[32]:
```
try:
    x,y = map(int, input().split())
    r = x / y
except Exception as e:
    print(f"!!Error!!{e}")
else:
    print("I will run if No Errors Come")
    print("Result: ", r)
finally:
    print("I will run!! Everytime Whatever happens")
```

```
4 5
I will run if No Errors Come
Result:  0.8
I will run!! Everytime Whatever happens
```

[37]:
```
try:
    x,y = map(int, input().split())
    r = x / y
except Exception as e:
    print(f"!!Error!!{e}")
else:
    print("I will run if No Errors Come")
    print("Result: ", r)
finally:
    print("I will run!! Everytime Whatever happens")
    print("We put here clean up actions")
```

```
1
!!Error!!not enough values to unpack (expected 2, got 1)
I will run!! Everytime Whatever happens
We put here clean up actions
```

[43]:
```
try:
    fp = open('filename', 'w')
    fp.write(b"asldkjf")

except Exception as e:
    print("!!Error!!", e)
```

```
else:
    print("content written sucessfully")
finally:
    fp.close()
    print("I will manage your resources whatever happens")
```

```
!!Error!! write() argument must be str, not bytes
I will manage your resources whatever happens
```

[42]:
```
fp.closed
```

[42]:
```
True
```

[45]:
```
def hello():
    try:
        1 / 0
        return 'not possible'
    except ZeroDivisionError as e:
        print("Error!!, Can not be divied by zero.")
        return 'ha ha ha'
    else:
        print("good good awesome awesome")
        return 'he he he'
    finally:
        return 'ho ho ho'
```

[46]:
```
ans = hello()
```

```
Error!!, Can not be divied by zero.
```

[47]:
```
print(ans)
```

```
ho ho ho
```

[56]:
```
def hello():
    try:
        int('alskdfj')
        1 / 0
        return 'not possible'
    except ZeroDivisionError as e:
        print("Error!!, Can not be divied by zero.")
        return 'ha ha ha'
    else:
        print("good good awesome awesome")
        return 'he he he'
    finally:
        # i will run whatever happens
```

```
        1 / 0 # causing
        return 'ho ho ho'
```

[57]:
```
hello()
```

```
      ␣
↪---------------------------------------------------------------------------

       ValueError                                Traceback (most recent call␣
↪last)

       <ipython-input-56-4c0614ca05f4> in hello()
         2     try:
   ----> 3         int('alskdfj')
         4         1 / 0


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


   During handling of the above exception, another exception occurred:


       ZeroDivisionError                         Traceback (most recent call␣
↪last)

       <ipython-input-57-a75d7781aaeb> in <module>
   ----> 1 hello()


       <ipython-input-56-4c0614ca05f4> in hello()
        12     finally:
        13         # i will run whatever happens
   ---> 14         1 / 0
        15         return 'ho ho ho'


       ZeroDivisionError: division by zero
```

[58]:
```
def hello():
    try:
        int('alskdfj')
        1 / 0
        return 'not possible'
    except ZeroDivisionError as e:
```

```
        print("Error!!, Can not be divied by zero.")
        return 'ha ha ha'
    else:
        print("good good awesome awesome")
        return 'he he he'
    finally:
        # i will run whatever happens
        #1 / 0 # causing
        return 'ho ho ho'
```

[59]:
```
hello()
```

[59]: 'ho ho ho'

[60]:
```
def hello():
    try:
        int('alskdfj')
        1 / 0
        return 'not possible'
    except ZeroDivisionError as e:
        print("Error!!, Can not be divied by zero.")
        return 'ha ha ha'
    else:
        print("good good awesome awesome")
        return 'he he he'
    finally:
        # i will run whatever happens
        #1 / 0 # causing
        print('ho ho ho')
```

[61]:
```
hello()
```

```
ho ho ho

      ␣
↪---------------------------------------------------------------------------

       ValueError                                Traceback (most recent call␣
↪last)

       <ipython-input-61-a75d7781aaeb> in <module>
   ----> 1 hello()


       <ipython-input-60-e7daf412a78a> in hello()
         1 def hello():
         2     try:
```

```
----> 3         int('alskdfj')
      4         1 / 0
      5         return 'not possible'


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

[50]:
```
def hi():
    try:
        int('alskdfj')
        1 / 0
        return 'not possible'
    except ZeroDivisionError as e:
        print("Error!!, Can not be divied by zero.")
        return 'ha ha ha'
    else:
        print("good good awesome awesome")
        return 'he he he'
```

[51]:
```
hi()
```

```
␣
↪---------------------------------------------------------------------------

ValueError                                Traceback (most recent call␣
↪last)

<ipython-input-51-ce8dd9c0d491> in <module>
----> 1 hi()


<ipython-input-50-6566a89c5b47> in hi()
  1 def hi():
  2     try:
----> 3         int('alskdfj')
  4         1 / 0
  5         return 'not possible'


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

0.0.2 Custom Exceptions

user defined error

use `raise` statement to `intentionally cause an exception` in our program

[62]:
```
raise MemoryError('ha ha my memory error')
```

```
      ␣
↪---------------------------------------------------------------------------

       MemoryError                               Traceback (most recent call␣
↪last)

       <ipython-input-62-1ce3b337a2f7> in <module>
   ----> 1 raise MemoryError('ha ha my memory error')


       MemoryError: ha ha my memory error
```

[63]:
```
raise PermissionError('ohh cool now i can make any error')
```

```
      ␣
↪---------------------------------------------------------------------------

       PermissionError                           Traceback (most recent call␣
↪last)

       <ipython-input-63-21f73010e43d> in <module>
   ----> 1 raise PermissionError('ohh cool now i can make any error')


       PermissionError: ohh cool now i can make any error
```

[64]:
```
raise ValueError()
```

```
      ␣
↪---------------------------------------------------------------------------

       ValueError                                Traceback (most recent call␣
↪last)

       <ipython-input-64-4954757c312d> in <module>
   ----> 1 raise ValueError()


       ValueError:
```

Write a code in python to take integer input

[66]:
```
while True:
    try:
        num = int(input("Enter integer: "))
    except ValueError:
        print("!!Error!!I said only integer do you get it ? ")
    else:
        print("Good Job!!")
        break
    finally:
        print("once more")
```

```
Enter integer: lskfd
!!Error!!I said only integer do you get it ?
once more
Enter integer: 123
Good Job!!
once more
```

write a code to take a `positive integer` input

[67]:
```
while True:
    try:
        num = int(input("Enter Positive Integer: "))
        if num <= 0:
            raise ValueError('I said only Positive Integer number')
    except ValueError as e:
        print(f"!!Error!! {e}")
    else:
        print("Good Job")
        break
```

```
Enter Positive Integer: -12
!!Error!! I said only Positive Integer number
Enter Positive Integer: 0
!!Error!! I said only Positive Integer number
Enter Positive Integer: -12
!!Error!! I said only Positive Integer number
Enter Positive Integer: -565
!!Error!! I said only Positive Integer number
Enter Positive Integer: 50
Good Job
```

`assert` statement is used to cause `AssertionError` based on `condition` also used in `testing`

syntax:

`assert cond, error messege`

```
    raise AssertionError if cond is False
```

[71]:
```
assert 1 > 2, '1 is not greater than 2 you idiot'
```

```
␣
↪---------------------------------------------------------------------------

AssertionError                            Traceback (most recent call␣
↪last)

<ipython-input-71-da400db7befa> in <module>
----> 1 assert 1 > 2, '1 is not greater than 2 you idiot'


AssertionError: 1 is not greater than 2 you idiot
```

[72]:
```
assert 2 > 1, '1 is not greater than 2 you idiot'
```

[75]:
```
while True:
    try:
        num = int(input("Enter Positive Integer: "))
        assert num > 0, 'I said only Positive Integer number'
    except AssertionError as e:
        print(f"!!Error!! {e}")
    else:
        print("Good Job")
        break
```

```
Enter Positive Integer: -10
!!Error!! I said only Positive Integer number
Enter Positive Integer: 0
!!Error!! I said only Positive Integer number
Enter Positive Integer: 10
Good Job
```

Custom Error Class

[80]:
```
class MyError(ValueError):
    def __init__(self, error_msg):
        super().__init__(error_msg)
        self.do_this()
        self.do_that()
    def do_this(self):
        print("\n\nHello I am do this an important precaustion should be done\
        to avoid any problem\n\n")
    def do_that(self):
```

```
        print("\n\nhello I am do that I do some stuff whenever MyError␣
↪Occurs\n\n")
```

[82]:
```
while True:
    try:
        num = int(input("Enter Positive Integer: "))
        if num <= 0:
            raise MyError('I said only Positive Integer number')
    except MyError as e:
        print(f"!!Error!! {e}")
    except ValueError as e:
        print(f"!!Error!! {e}")
    else:
        print("Good Job")
        break
```

```
Enter Positive Integer: -10


Hello I am do this an important precaustion should be done        to avoid any
problem



hello I am do that I do some stuff whenever MyError Occurs


!!Error!! I said only Positive Integer number
Enter Positive Integer: 10
Good Job
```

[90]:
```
class MyError(ValueError):
    def __init__(self, error_msg):
        self.error_msg = error_msg
        self.do_this()
        self.do_that()
    def do_this(self):
        print("\n\nHello I am do this an important pre-caustion should be done\
        to avoid any problem\n\n")
    def do_that(self):
        print("\n\nhello I am do that I do some stuff whenever MyError␣
↪Occurs\n\n")
    def __str__(self):
        return self.error_msg
```

[91]:
```
try:
    raise MyError("this is the internal workflow")
except MyError as e:
    print(e)
```

```
Hello I am do this an important precaustion should be done        to avoid any
problem



hello I am do that I do some stuff whenever MyError Occurs

this is the internal workflow
```

[92]:
```
raise MyError('ha ha ha')
```

```
Hello I am do this an important precaustion should be done        to avoid any
problem



hello I am do that I do some stuff whenever MyError Occurs
```

```
      ␣
↪---------------------------------------------------------------------------

       MyError                                   Traceback (most recent call␣
↪last)

       <ipython-input-92-d868e0bbd4c3> in <module>
   ----> 1 raise MyError('ha ha ha')

       MyError: ha ha ha
```

```
Let me remind if some topics of python has remained
```

0.0.3 Advance Python

tkinter application development

Atom Editior

[ ]: