Lecture 28
Exception

OOP – Spring 2022 (Python)

Exception Example-1

```
print (2/0)
```

```
Traceback (most recent call last):
File "e:\subjects\spring 2022\oop_python\class exercise\lecture 28\exception_1.py", line 1, in <module>
print (2/0)
ZeroDivisionError: division by zero
```

Exception Example-2

```
abc=20
print(abd)
```

```
Traceback (most recent call last):
File "e:\subjects\spring 2022\oop_python\class exercise\lecture 28\exception_1.py", line 3, in <module>
print(abd)
NameError: name 'abd' is not defined
```

Exception Example-3

```
print('Welcome)
```

```
File "e:\subjects\spring 2022\oop_python\class exercise\lecture 28\exception_1.py", line 4
print('Welcome)
^
SyntaxError: EOL while scanning string literal
>>>
```

Exception Example-4

```
mydict = {}

mydict['x']
```

```
Traceback (most recent call last):
File "<pyshell#15>", line 1, in <module>
mydict['x']
KeyError: 'x'
```

Exception Example-5

```
values = [1,2,3,4]
values[5]
```

```
Traceback (most recent call last):
File "<pyshell#20>", line 1, in <module>
values[5]
IndexError: list index out of range
```

Catching/Trapping Exception

Unusual but expected error

try and except are used to catch exceptions

on error (try block), Python match except block to handle

```
try:
    some statement here
except:
    exception handling
```

Exception – Without Try/ Catch

```
number = int(input('Enter the number 1-10:'))
print('You have entered number', number)
```

```
Please enter the number between 1 & 10: 'Hi'
Traceback (most recent call last):
File "C:/Python/Python361/exception2.py", line 1, in
<module>
number = int(input('Please enter the number between 1 &
10: '))
ValueError: invalid literal for int() with base 10: "'Hi'"
     some statement here
```

Exception Example – Without Try/ Catch

```
number = int(input('Enter the number 1-10:'))
print('You have entered number', number)
```

```
Please enter the number between 1 & 10: 'Hi'
Traceback (most recent call last):
File "C:/Python/Python361/exception2.py", line 1, in
<module>
number = int(input('Please enter the number between 1 &
10: '))
ValueError: invalid literal for int() with base 10: "'Hi'"
     some statement here
```

Exception Example – With Try/ Catch

```
try:
    number = int(input('Enter number 1-10: '))
    print('You have entered number: ',number)
except(ValueError):
    print('Error..numbers only')print('You have
entered number: ',number)
```

```
Enter number 1-10: g
Error..numbers only
```

Raise Exception

raise/ throw an exception, use raise keyword

```
raise ExceptionClass('Some Text Here')
```

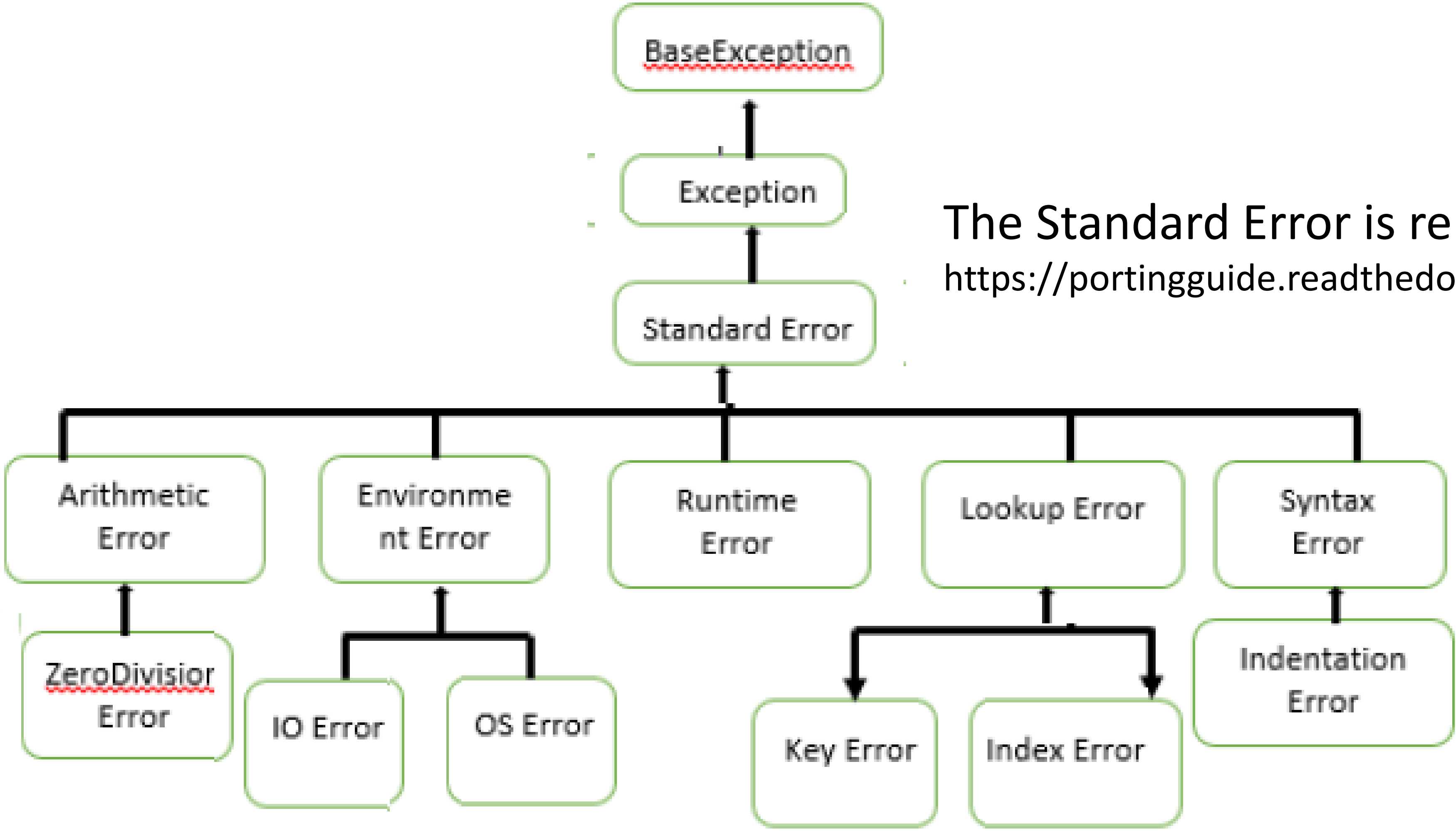
Exception Example – With Raise

```python
def enterAge(age):
    if age<0:
        raise ValueError('Only positive integers are allowed')
    if age % 2 ==0:
        print('Entered Age is even')
    else:
        print('Entered Age is odd')
try:
    num = int(input('Enter your age: '))
    enterAge(num)
except ValueError:
    print('Only positive integers are allowed')
```

```
Enter your age: -5
Only positive integers are allowed
```

Creating Custom exception class

We can create a custom exception class by Extending BaseException class or subclass of BaseException.



The Standard Error is removed from Python 3.0

https://portingguide.readthedocs.io/en/latest/exceptions.html

https://docs.python.org/3/library/exceptions.html

Custom Exception Example

```
class NegativeNumberException(RuntimeError):
    def __init__(self, age):
        super().__init__()
        self.age = age
```

Custom Exception Test Example

```
from negative_number_exception import *
  def enterage(age):
   if age < 0:
      raise NegativeNumberException('Negative integers not allowed')
   if age % 2 == 0:
     print('Age is Even')
   else:
     print('Age is Odd')
try:
    num = int(input('Enter your age: '))
    enterage(num)
except NegativeNumberException:
    print('Only positive integers are allowed')
except:
    print('Something is wrong')
```

Custom Exception Example Test Run

```
Enter your age: 28
Age is Even

Enter your age: -25
Only positive integers are allowed

Enter your age: small
Something is wrong
```