Assignment 12: 12 Feb 2023

Q1. What is an Exception in python? Write the difference between Exceptions and Syntax errors.
Q2. What happens when an exception is not handled? Explain with an example.
Q3. Which Python statements are used to catch and handle exceptions? Explain with an example.
Q4. Explain with an example:
a. try and else
b. finally
c. raise
Q5. What are Custom Exceptions in python? Why do we need Custom Exceptions? Explain with an example.
Q6. Create a custom exception class. Use this class to handle an exception.

Q1. What is an Exception in python? Write the difference between Exceptions and Syntax errors.

Ans: An Exception in python is an error that happens during the execution of a program. Whenever there is an error, Python generates an exception that could be handled. It basically prevents the program from getting crashed.

S.No.	Exception errors	Syntax errors
1	It is raised when some internal events occur which changes the normal flow of the program.	it is caused by the wrong syntax in the code. It leads to the termination of the program.
2	Exception can work in developer system but not in user system	Syntax Error is an absolute error by the developer
3	When in the runtime an error that occurs after passing the syntax test is called exception or logical type.	occurs when the interpreter encounters invalid syntax in code
4	Eg- IndexError, AssertionError, AttributeError, ImportError, KeyError, NameError, MemoryError & TypeError	If the interpreter finds any invalid syntax during the parsing stage, a SyntaxError is thrown.

Note- When errors are raised then we handle that with the help of the Handling method.

Q2. What happens when an exception is not handled? Explain with an example.

Ans: If an exception occurs during execution of the try clause, the exception may be handled by an except clause. If the exception is not handled by an except clause, the exception is re-raised after the finally clause has been executed.

 # Example of exception

```
# initialize the amount variable
marks = 10000
```

```
# perform division with 0
a = marks / 0
print(a)
```

Output

```
ZeroDivisionError: division by zero
```

Q3. Which Python statements are used to catch and handle exceptions? Explain with an example.

Ans: The try and except code block in Python is used to catch and handle exceptions.

```
#example
try:# try for error
  1/0
except Exception as e: # if such error comes send it to except block
   print("This is Except Block::: ", str(e).title())
```

output

```
This is Except Block:::  Division By Zero
```

Q4. Explain with an example:

a. try and else

b. finally

c. raise

Ans: There are 4 blocks :

1. i. try
    a. try for error
2. ii. except
    a. if error comes it will come to except block
3. iii. else
    a. if try block executed without any error then else will be executed for residual code, else will not executed if try fails
4. iv. finally
    a. it will be executed at any situation

Example are given below

#a.1)try-except example

```
try:# try for error
   1/0
except Exception as e: # if such error comes send it to except block
   print("This is Except Block::: ", e)
```

output

```
This is Except Block:::  division by zero
```

 # a.2)else:

```
try:
   a=[]
   for i in range(0,10):
      if i<=0:
         a.append(0)  # we have not printed the list a here

      else:
         a.append(round(1/i,3))
```

```
except Exception as e:
    print (str(e).capitalize())
else:
    print(a)
    print()
    print('Exception Handled')
```

output

```
[0, 1.0, 0.5, 0.333, 0.25, 0.2, 0.167, 0.143, 0.125, 0.111]
Exception Handled
```

#b) finally

```
try:
    a=[]
    for i in range(0,10):
            a.append(1/i)
    except Exception as e:
    print (f'There is a bug of : {str(e).title()}')
else:
    print(a)
    print()
    print('Exception Handled')
finally:
    print()
    print("FINALLY "*5) # it will be executed at any situation
```

output

```
There is a bug of : Division By Zero
FINALLY FINALLY FINALLY FINALLY FINALLY
```

#c) raise

```
a = 5
if a % 2 != 0:
        raise Exception("The number shouldn't be an odd integer")
```

output

```
Exception: The number shouldn't be an odd integer
```

Q5. What are Custom Exceptions in python? Why do we need Custom Exceptions? Explain with an example.

Ans: In Python, we can define custom exceptions by creating a new class that is derived from the built-in Exception class.
Built-in exceptions offer information about Python-related problems, and custom exceptions will add information about project-related problems.
Example:

```
# according to python the value for age is fine but not for you
age= int(input("ENTER YOUR AGE::::")) # enter age -111
print(age)


# to handle this kind of exception we need to create a subclass of Exception super class
```

output

```
ENTER YOUR AGE::::-111
```

-111

Q6. Create a custom exception class. Use this class to handle an exception.

Ans:

```
class Validateage(Exception): # the subclass where super class is Exception
    def __init__(self, msg):
        self.msg=msg
# make and function to raise error
def validateage(age):
    if age<0:
        raise Validateage(f"Age can't be Negative {age} ")
    elif age>=200:
        raise Validateage(f"Age can't be too heigh  {age}")
    else:
        print(f"This is a Valid Age  {age}")
# Custome Exception Handelling
try:
    age= int(input("ENTER YOUR AGE::::"))
    validateage(age)
except Validateage as v:
    print("There is a Bug as:::::", v)
else:
    print()
    age1=age*2
    print(f"Your age  will be double to {age1} in next {age} years")
ENTER YOUR AGE::::11
This is a Valid Age  11
Your age  will be double to 22 in next 11 years
```