Batch: A3 Roll No.: 1911034

Experiment / assignment / tutorial No.___5____

Grade: AA / AB / BB / BC / CC / CD /DD

Signature of the Staff In-charge with date

TITLE: To implement User Defined Exceptions in Python

AIM: To understand the concept of user defined exceptions with Queue Data Structure.

Expected OUTCOME of Experiment:

CO - CO3

Books/ Journals/ Websites referred:

1. Geeksforgeeks.com
2. W3Schools.com
3. Tutorialspoint.com
4. Ma'am's ppt

Pre Lab/ Prior Concepts:

What is Exception?

An exception is an event, which occurs during the execution of a program that disrupts the normal flow of the program's instructions. In general, when a Python script encounters a situation that it cannot cope with, it raises an exception. An exception is a Python object that represents an error.

When a Python script raises an exception, it must either handle the exception immediately otherwise it terminates and quits.

Handling an exception

If you have some *suspicious* code that may raise an exception, you can defend your program by placing the suspicious code in a **try:** block. After the

try: block, include an **except:** statement, followed by a block of code which handles the problem as elegantly as possible.

Syntax

Here is simple syntax of *try....except...else* blocks −

```
try:
   You do your operations here;
   ......................
except ExceptionI:
   If there is ExceptionI, then execute this block.
except ExceptionII:
   If there is ExceptionII, then execute this block.
   ......................
else:
   If there is no exception then execute this block.
```

Here are few important points about the above-mentioned syntax −

- A single try statement can have multiple except statements. This is useful when the try block contains statements that may throw different types of exceptions.
- You can also provide a generic except clause, which handles any exception.
- After the except clause(s), you can include an else-clause. The code in the else-block executes if the code in the try: block does not raise an exception.
- The else-block is a good place for code that does not need the try: block's protection.

Problem statement:

User Defined Exception
Write a Program that implements a Queue data structure of specified size.if the queue becomes full and we still try to add an element to it then a user-defined **QueueError** exception should be raised.similarly, if the queue is empty and we try to delete an element from it then a **QueueError** exception should be raised .

Program (with comments) (Write comments to your program to explain the implementation)

```
class QueueError(Exception): #user defined exception class
    def __init__(self,msg):
        self.msg=msg

rear = -1
front = 0
maxsize = 5
queue = list()
def checkqueue(element,choice): #function that checks whether addition or
 deletion from queue is possible, if not it raises an error
    global rear, front, maxsize , queue
    if front==0 and rear==maxsize-
1 and choice==1: #theseare the conditions to check if the queue is full
        raise QueueError("The queue is already full. Cannot add more elem
ents")
    elif choice==2 and front>rear: #conditions to check if queue is empty
        raise QueueError("The queue is empty. Cannot delete more elements
")
    else:
        changequeue(element,choice)


def changequeue(element,choice): #function to append or delete elements f
rom a queue
    global rear, front,maxsize,queue
    if choice==1:
        rear = rear+1
        queue.append(element)

        print("element added is ",queue[rear])
    elif choice==2:
        print("element deleted is ",queue[front])
        front=front+1
```

```
def displayqueue():
    global front , rear , queue
    for i in range (front,rear+1):
        print(" ",queue[i])

def initialize():
    var=True
    while var==True:
        choice = int(input("Enter 1 to add an element, 2 to delete an ele
ment or 3 to display the queue "))
        if choice==1:
            element = int(input("Enter the element to be added to the que
ue"))
            try:
                checkqueue(element,1)
            except QueueError as sizefull: #catching the exception thrown
 by checkqueue() function
                print(sizefull)
        elif choice==2:
            print("Trying to delete ")
            try:
                checkqueue(-1,2)
            except QueueError as sizenull: #catching the exception thrown
 by checkqueue() function
                print(sizenull)
        else:
            displayqueue()


initialize()
```

Output Screenshots:

The max size of the queue is 5 , so if we try to add more than five elements it displays a queue error.

```
Enter 1 to add an element, 2 to delete an element or 3 to display the queue 1
Enter the element to be added to the queue21
element added is  21
Enter 1 to add an element, 2 to delete an element or 3 to display the queue 1
Enter the element to be added to the queue22
element added is  22
Enter 1 to add an element, 2 to delete an element or 3 to display the queue 1
Enter the element to be added to the queue23
element added is  23
Enter 1 to add an element, 2 to delete an element or 3 to display the queue 1
Enter the element to be added to the queue24
element added is  24
Enter 1 to add an element, 2 to delete an element or 3 to display the queue 1
Enter the element to be added to the queue25
element added is  25
```

```
Enter 1 to add an element, 2 to delete an element or 3 to display the queue 3
  21
  22
  23
  24
  25
Enter 1 to add an element, 2 to delete an element or 3 to display the queue 1
Enter the element to be added to the queue26
The queue is already full. Cannot add more elements
```

After deleting all five elements if we try to delete any more it will display a queue error:

```
Enter 1 to add an element, 2 to delete an element or 3 to display the queue 2
Trying to delete
element deleted is  21
Enter 1 to add an element, 2 to delete an element or 3 to display the queue 2
Trying to delete
element deleted is  22
Enter 1 to add an element, 2 to delete an element or 3 to display the queue 2
Trying to delete
element deleted is  23
Enter 1 to add an element, 2 to delete an element or 3 to display the queue 2
Trying to delete
element deleted is  24
Enter 1 to add an element, 2 to delete an element or 3 to display the queue 2
Trying to delete
element deleted is  25
Enter 1 to add an element, 2 to delete an element or 3 to display the queue 2
Trying to delete
The queue is empty. Cannot delete more elements
```

Conclusion: In this experiment we have understood how to implement user defined exceptions with the help of queue data structure.

Date: _____ **Signature of faculty in-charge**