implement python generator and decorators
TASK 8. Date: 17-9-25

Aim:
To write a python program to implement python generator and decorator.

8.1 write a python program that includes a generator function to produce a sequence of numbers The generator should be able to.

a. produce a sequence of numbers

b. produce a default sequence of numbers starting from 0, ending at 10,

Algorithm:

1. Define Generator function

2. initialize current value

3. Generate sequence

4. Get user input

5. Generate iterable object.

6. print Generated sequence

program.

```
def number_sequence(start, end, step=1):
        current = start
        while current <= end:
          yield current
          current += sleep
start = int(input("Enter starting number:"))
```

OUTPUT

```
0
1
2
```

```
end = int(input("enter ending number:"))
step = int( input("enter step value:"))
sequence_generator = sequence of numbers
for number in sequence_generator:
    print (number).
```

8.1(b)

Program:

```
def my generator(n):
 value =0
 while value<n:
       yield value
        value += 1
for value in my generator(3):
 4  print(value)
```

TASK 8.2

Imagine you are working on a messaging applica
that needs to format messages differently based
on user's preferences. user can choose to have
their message automatically converted to
uppercase or to lowercase. you are provided with two
decorators: uppercase-decorator, and lower case-decorat
These decorators modify the behaviour of the
function. they decorate by converting the text
to uppercase or lower case, respectively. write a
program to implement it.

Algorithm:

1. Create Decorators
2. Define Function
3. Define Greet Function
4. Execute the program

program:

```
def uppercase_decorator (func):
    def wrapper (text):
        return func(text).upper()
    return wrapper

def lowercase_decorator (func):
    def wrapper (text):
        return func (text).lower()
```

OUTPUT

```
HI, I AM CREATED BY A FUNCTION PASSED
AS AN ARGUMENT.
hi, Iam created by a function passed as an
argument
```

```
return wrapper
@uppercase_decorator
def shout(text):
    return text
@lowercase_decorator
def whisper(text):
    return text
def greet(func):
    greeting = func("Hi, I am created by a function
                          passed as an argument")
    print(greeting)
greet(shout)
greet(whisper)
```

VEL TECH - CSE	
EX NO.	8
PERFORMANCE (5)	5
RESULT AND ANALYSIS (3)	3
VIVA VOCE (3)	3
RECORD (4)	4
TOTAL (15)	15
SIGN WITH DATE	

Result:

Thus the python program to implement python generator and decorator was successfully executed output was verified

implement Exceptions and Exceptional handling
in python

TASK9

Date: 15-10-25

Aim: To implement Exceptions and Exceptional handling in python.

Q.1 you are developing a python program that process a list of students grades the program is designed to allow the user to select a grade by specifying an index number. However, you need to ensure that program handles cases where the user inputs an index that is out of range, i.e, an index that does not exist in the list

Algorithm:

1. Start
2. Initialize a list of grades
3. prompt user to enter index of the grade they want to
4. Attempt to display grade at specified index
5. If index is out of range, catch IndexError and print an error message.

program:

```
grades = [85, 90, 78, 92, 88]
print("Grades List:", grades)
try:
    index = int(input("Enter index of grade you wa
                       to view:"))
    print(f"The grade at index {index}:
           {grades[index]}")
```

OUTPUT

```
Grades List: [85,90,78,92,88]
Enter index of grade you want to view: 10
Invalid index. please enter a valid index
```

```
except indexError:
    print ("invalid index. please enter a valid index
```

TASK 9.2

you are developing Python calculator program that performs basic arithmetic operations. one of the key functionalities is to divide two numbers entered by the user. However, dividing by zero is not allowed and would cause the program to crash if not handled properly.

Algorithm:

. Start

2. prompt the user to enter two number.

3. Attempt to divide numerator by denominator

4. If the denominator is zero, catch the zero Division error and displays an error message "Error: Division by zero is not allowed."

Program:

```
def divide_numbers ():
 try:
   numerator = float(input("Enter numerator:"))
   denominator = float(input("Enter Denominator:"))
   results = numerator / Denominator.
   print(f" Result:{result}")
except zero DivisionError:
    print("Error: Division by zero is not allowed")
```

Output

```
Enter a number: 15
Exception occured : Invalid Age
```

```
except value error:
    print("error: please. enter valid numbers!")
divide_numbers()
```

VEL TECH - CSE	
EX NO.	9
PERFORMANCE (5)	5
RESULT AND ANALYSIS (3)	3
VIVA VOCE (3)	3
RECORD (4)	4
TOTAL (15)	
SIGN WITH DATE	15

Result:

Thus the python program for implement exceptions and exceptional handling is executed and verified successfully