

Implement Python generator and decorator

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 include 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 default object.

6. Print generated sequence

Program:

```
def number_sequence(start, end, step=1):
```

```
    current = start
```

```
    while current <= end:
```

```
        yield current
```

```
        current += step
```

```
start = int(input("Enter starting number:"))
```

Output

0

2

~~Output = 2000~~

~~Output = 2000~~

~~Output = 2000~~

~~Output = 2000~~

~~Output = 2000~~

~~Output = 2000~~

~~Output = 2000~~

~~Output = 2000~~

~~Output = 2000~~

~~Output = 2000~~

~~Output = 2000~~

~~Output = 2000~~

~~Output = 2000~~

~~Output = 2000~~

~~Output = 2000~~

~~Output = 2000~~

~~Output = 2000~~

~~Output = 2000~~

~~Output = 2000~~

~~Output = 2000~~

~~Output = 2000~~

~~Output = 2000~~

```
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
```

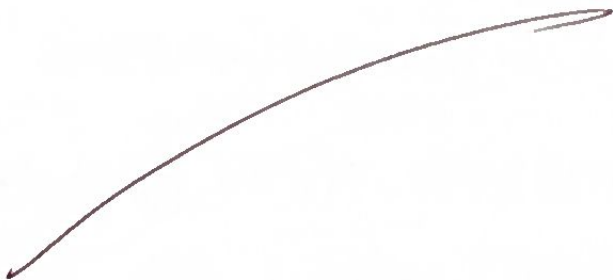
```
    while value < n:
```

```
        yield value
```

```
        value += 1
```

```
for value in my_generator(5):
```

```
    print(value)
```



TASK 8.2

Imagine you are working on a messaging application 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 lowercase-decorator. These decorators modify the behaviour of the function. They decorate by converting the text to uppercase or lowercase respectively. Write a program to implement it.

Algorithm:

1. Create Decorators
2. Define Function
3. Define Decorator 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, i am created by a function passed as an
argument

shout wrapper.

@upper case decorator

def shout (text):

return text

@lower case 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 - COE	
FX NO.	P
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 to implement python
generator and decorator was successfully
executed and output was verified

Implement Exceptions and Exceptional handling in Python

Task 19

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 student 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 to view

4. Attempt to display grade at specified index

5. If index is out of range, catch index error 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 want to view:"))
```

```
print(f"The grade at index {index} is {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:

1. Start
2. Prompt the user to enter two numbers.
3. Attempt to divide numerator by denominator.
4. If the denominator is zero, catch the zero division error and display 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: "))  
        result = numerator / denominator  
        print(f"Result: {result}")  
    except ZeroDivisionError:  
        print("Error: Division by zero is not allowed.")
```

Output

Enter a number: 15

Exception occurred: Invalid Age

except ValueError:

```
    print("Error: Please enter valid number!\n")  
divide_number())
```

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 executed and verified successfully