# Various helper functions

def print\_header(section: str):

print('--------------------------')

print('Start of section ' + section)

def print\_footer(section: str):

print('End of section ' + section)

1)Assign the value 7 to the variables guess\_me .then write the conditional tests(if,else,elif) to print the string 'too low' if guess\_me is less than 7,'too high' if greater than 7 and' just right ' if equal to 7

Ans.

print\_header(section)

guess\_me = 7

if guess\_me < 7:

print("too low")

elif guess\_me > 7:

print("too high")

else:

print("just right")

print\_footer(section)

2)Assign the value 7 to the variable guess\_me and the value 1 to the variable start.write a while loop that compares start with guess\_me .print too low if start is less than guess me .if starts equals guess\_me,print 'found\_it!' and exit the loop .if start is greater than guess\_me,print the 'oops' and exit the loop.increment start at the end of the loop.

Ans :

print\_header(section)

guess\_me = 7

start = 1

while True:

if start < guess\_me:

print('too low')

elif start == guess\_me:

print('found it!')

break

else:

print('oops')

break

start += 1

print\_footer(section).

3)print the following values of the list [3,2,1,0] using a for loop.

Ans :

print\_header(section)

numbers = [3, 2, 1, 0]

for num in numbers:

print(num)

print\_footer(section)

4)use a list comprehension to make a list of the even number in range[10]

Ans :

print\_header(section)

even\_numbers = [num for num in range(10) if num % 2 == 0]

print(even\_numbers)

print\_footer(section)

5)use a dictionary comprehension to create the dictionary squares.use range [10] to return the keys and use the square of each key as its value.

Ans:

print\_header(section)

limit = 10

squares = {num: num \* num for num in range(limit)}

print(squares)

print\_footer(section)

6)construct the set odd from the odd numbers in the range using a set comprehension [10].

Ans :

print\_header(section)

limit = 10

odd = {num for num in range(limit) if num % 2 == 1}

print(odd)

print\_footer(section)

7)use a generator comprehension to return string 'Got' and a number for the numbers in range [10].iterate through this by using a for loop.

Ans :

print\_header(section)

limit = 10

string\_generator = ('Got ' + str(num) for num in range(limit))

for item in string\_generator:

print(item)

print\_footer(section)

8)define the function called good that returns the list ['harry','ron','hermione'].

Ans :

print\_header(section)

def good():

return ['Harry', 'Ron', 'Hermione']

print(good())

print\_footer(section)

9)define a generator function called gets\_odds that returns the odd number from range[10].use a for loop to find and print the third value returned.

Ans :

print\_header(section)

limit = 10

get\_odds = (num for num in range(limit) if not num % 2 == 0)

count = 0

for num in get\_odds:

if count == 2:

print(num)

break

count += 1

print\_footer(section)

def test(func):

def nested\_function(\*args, \*\*kwargs):

print('start')

result = func(\*args, \*\*kwargs)

print('end')

return result

return nested\_function

@test

def add(a, b):

print(a + b)

add(12,4)

10)define an exception called oops exception,raise this exception to see what happens,the write the code to catch this exception & print 'caught an oops'.

Ans :

print\_header(section)

class OopsException(Exception):

pass

def with\_exception(a):

if a < 0:

raise OopsException(a)

try:

with\_exception(-1)

except OopsException as err:

print('Caught an oops')

print\_footer(section)

11)use zip() to make a dictionary called movies that pairs these lists:titles =['creature of habit','crewel fate'] and plots =['A nun turns into a monster','A haunted yarn shop']

Ans :

print\_header(section)

titles = ['Creature of Habit', 'Crewel Fate']

plots = ['A nun turns into a monster', 'A haunted yarn shop']

movies = {}

for title, plot in zip(titles, plots):

movies[title] = plot

# or movies = dict(zip(titles, plots))

print(movies)

print\_footer(section)