1. Assign the value 7 to the variable guess\_me. Then, write the conditional tests (if, else, and 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.

# assign the value 7 to the variable guess\_me. Then, write the conditional tests (if, else, and 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.

guess\_me = 7

if guess\_me < 7:

print('too low')

elif guess\_me > 7:

print('too high')

elif guess\_me == 7:

print('just right')

1. 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 start equals guess\_me, print 'found it!' and exit the loop. If start is greater than guess\_me, print 'oops' and exit the loop. Increment start at the end of the loop.

guess\_me = 7

start = 1

while start <= guess\_me:

# print 'too low' if start is less than or equal to guess\_me

print('too low')

if start == guess\_me:

# print 'found it!" if start is equal to guess\_me

print('found it!')

break

if start > guess\_me:

# print 'oops' if start is greater than guess\_me

print('oops')

break

start = start + 1

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

# Print the following values of the list [3, 2, 1, 0] using a for loop

# define the list

l1 = [3, 2, 1, 0]

# use for loop to iterate through the list and print it

for i in l1:

print(i)

1. Use a list comprehension to make a list of the even numbers in range(10)

# Use a list comprehension to make a list of the even numbers in range(10)

# declare a list l2

l2 = []

# use the following list comprehension to to make a list of the even numbers in the range 0 to 10

l2 = [i for i in range(11) if i % 2 == 0]

print("Print the list containing even numbers")

print(l2)

1. 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.

# 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

# Declare a dictionary

dict1 = {}

# use the following dictionary comprehension to create the dictionary squares.

dict1 = {num : num\*num for num in range(1,11)}

print("Print the dictionary")

print(dict1)

1. Construct the set odd from the odd numbers in the range using a set comprehension (10).

# Construct the set odd from the odd numbers in the range using a set comprehension (10)

# declare a set set1

set1 = ()

# use the following set comprehension to make a set of the odd numbers in the range 0 to 10

set1 = {i for i in range(11) if i % 2 != 0}

print("Print the set containing add numbers")

print(set1)

1. Use a generator comprehension to return the string 'Got ' and a number for the numbers in range(10). Iterate through this by using a for loop.

# Use a generator comprehension to return the string 'Got ' and a number for the numbers in range(10).

# Iterate through this by using a for loop.

# Create a generator comprehension to return the string 'Got ' and a number for the numbers in range(10).

Gen1 = ("Got"+str(i) for i in range(11))

# print each item of the generator Gen1

for i in Gen1:

print(i)

1. Define a function called good that returns the list ['Harry', 'Ron', 'Hermione'].

# Define a function called good that returns the list ['Harry', 'Ron', 'Hermione'].

def good():

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

# call the function which returns the list

l1 = good()

print(l1)

1. Define a generator function called get\_odds that returns the odd numbers from range(10). Use a for loop to find and print the third value returned.

# Define a generator function called get\_odds that returns the odd numbers from range(10).

# Use a for loop to find and print the third value returned.

# Define a generator function GenFunction1()

def GenFunction1():

for i in range(0,10):

if i%2 != 0:

yield i

# create an object of the generator GenFunction1()

obj = GenFunction1()

# use for loop to find and print the third value returned

for i in obj:

if next(obj).\_\_index\_\_() == 3:

print(next(obj))

1. Define an exception called OopsException. Raise this exception to see what happens. Then write the code to catch this exception and print 'Caught an oops'.

# using custom exception in python

# Declare a custom exception OopsException

# All exceptions in python inherit from class Exception

class OopsException(Exception):

pass

try:

raise OopsException("some message")

except OopsException as e:

print(e)

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'].

# 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'].

# import defaultdict from collections package

from collections import defaultdict

# declare two lists provided

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

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

# create an object of defaultdict(list) class

ddict1 = defaultdict(list)

# store lists titles, plots into the defaultdict object

for iteml1, iteml2 in zip(titles, plots):

ddict1[iteml1].extend(iteml2)

# create a dictionary object and store the value

dict1 = dict(ddict1)

print(type(dict1))

# Display dictionary object key, value pairs

print(dict1)