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.

**Answer:**

**guess\_me = 7**

**if guess\_me < 7:**

**print("too low")**

**elif guess\_me > 7:**

**print("too high")**

**else:**

**print("just right")**

**🡪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.

**Answer:**

**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**

**🡪**

**too low**

**too low**

**too low**

**too low**

**too low**

**too low**

**found it!**

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

**Answer:**

**list\_1 = [3,2,1,0]**

**for i in list\_1:**

**print(i)**

**🡪**

3

2

1

0

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

**Answer:**

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

**print(even\_number)**

**🡪**

[0, 2, 4, 6, 8]

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.

**Answer:**

**squares = {num: num \* num for num in range(10)}**

**print(squares)**

**🡪**

{0: 0, 1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81}

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

**Answer:**

**odd = {num for num in range(10) if num % 2 == 1}**

**print(odd)**

🡪 {1, 3, 5, 7, 9}

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.

**Answer:**

**string\_generator = ('Got ' + str(num) for num in range(10))**

**for item in string\_generator:**

**print(item)**

**🡪**

Got 0

Got 1

Got 2

Got 3

Got 4

Got 5

Got 6

Got 7

Got 8

Got 9

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

**Answer:**

**def good():**

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

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.

**Answer:**

**get\_odds = (num for num in range(10) if not num % 2 == 0)**

**count = 0**

**for num in get\_odds:**

**if count == 2:**

**print(num)**

**break**

**count += 1**

**🡪5**

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'.

**Answer:**

**class OopsException(Exception):**

**pass**

**def with\_exception(a):**

**if a < 0:**

**raise OopsException(a)**

**try:**

**with\_exception(-2)**

**except OopsException as err:**

**print('Caught an oops')**

**🡪** Caught an oops

1. 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'].

**Answer:**

**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**

**print(movies)**

🡪 {'Creature of Habit': 'A nun turns into a monster', 'Crewel Fate': 'A haunted yarn shop'}

In [ ]: