1. Give the variable guess me the value 7. Then, if guess me is less than 7, write the conditional tests (if, else, and elif) to print the string 'too low', 'too high', and 'just right' if guess me is greater than 7, and 'just right' if guess me is equal to 7.

Ans:- guess\_me = 7

if guess\_me < 7:

print(‘too low’)

elif guess\_me > 7:

print(‘too high’)

else:

print(‘just right’)

2. Give the variable guess me the value 7 and the variable start the value 1. Create a while loop that compares the start and guess me variables. If the start is less than guess me, the print will be too tiny. Print 'found it!' and then exit the loop if start equals guess me. Print 'oops' and exit the loop if start is greater than guess me. The increments begin at the loop's end.

Ans:- guess\_me = 7

Start = 1

While True:

if start < guess\_me :

print(‘too tiny’)

elif start = = guess\_me :

print(‘found it!’)

break

else :

print(‘oops’)

break

start =start+1

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

Ans:- for i in [3,2,1,0]:

print(i)

OR

list = [3,2,1,0]

for i in list:

print(i)

4. Make a list of even numbers in the range using list comprehension (10).

|  |
| --- |
| Ans:- |

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

print(even\_numbers) **Output:-**

[2, 4, 6, 8]

5. To make the dictionary squares, use a dictionary comprehension. To return the keys, use range(10) and the square of each key as the value.

Ans:-

# dictionary comprehension example

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

print(square\_dict)

output:-

{1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81, 10: 100}

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

Ans:-

|  |
| --- |
| list = [1, 2, 3, 4,5,6,7,8,9]    set\_using\_comp = { number for number in range(1, 11) if number % 2 != 0}    print("Output Set using set comprehensions:",                                set\_using\_comp)  output:- Output Set using set comprehensions: {1, 3, 5, 7, 9} |

7. Return the string 'Got'and a number for the numbers in range using a generator comprehension (10). Using a for loop, iterate through this.

Ans:-

|  |
| --- |
| limit = 10 |
|  |

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

|  |
| --- |
| for item in string\_generator: |
|  |

|  |
| --- |
| print(item) |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

8. Create a good feature that returns the list ['Harry, Ron, and Hermione'].

Ans:-

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | def good(): | |  |  |  | | --- | | return ['Harry', 'Ron', 'Hermione'] | |  |  |  | | --- | |  | |  |  |  | | --- | |  | |  |  |  | | --- | | print(good()) | |  | |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

9. Create a get odds generator function that returns the odd numbers from a set (10). To find and print third value returned, use a for loop.

Ans:-

|  |
| --- |
| 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 |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| # def get\_odds(): |
|  |

|  |
| --- |
| # for number in range(1, 10, 2): |
|  |

|  |
| --- |
| # yield number |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| # count = 1 |
|  |

|  |
| --- |
| # for number in get\_odds(): |
|  |

|  |
| --- |
| # if count == 3: |
|  |

|  |
| --- |
| # print("The third odd number is", number) |
|  |

|  |
| --- |
| # break |
|  |

|  |
| --- |
| # count += 1 |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
| Output:- 5  The third odd number is 5 |

10. Create a test decorator that prints'start' when a feature is called and 'end' when it is completed.

Ans:-

|  |
| --- |
| 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)

11. Create the OopsException exception. To see what happens, raise this exception. Then write the code to catch the exception and show 'Caught an oops'.

Ans:-

|  |
| --- |
| 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')

12. Create a dictionary called movies using zip() that pairs these lists: titles = ['Crewel Fate,' 'Creature of Habit'] ['A nun transforms into a monster,' 'A haunted yarn shop,'] and plots =

Ans:-

|  |
| --- |
| 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)

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