**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.**

guess\_me = 7

if guess\_me < 7:

print("too low")

elif guess\_me > 7:

print("too high")

else:

print("just right")

**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 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:

if start < guess\_me:

print("too low")

elif start > guess\_me:

print("oops")

break

start += 1

print("found it!")

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

for number in [3, 2, 1, 0]:

print(number)

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

Here is the code in Python to make a list of even numbers in the range 0 to 10 using a list comprehension:

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

print(even\_numbers)

This code uses a list comprehension to construct a new list called even\_numbers. The list comprehension [x for x in range(10) if x % 2 == 0] generates a list of values x taken from the range 0 to 10 that satisfy the condition x % 2 == 0, i.e. it generates a list of even numbers. The print statement then outputs the resulting list of even numbers, which is [0, 2, 4, 6, 8].

**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.**

Here is the code in Python to create a dictionary called squares using a dictionary comprehension:

squares = {x: x\*\*2 for x in range(10)}

print(squares)

This code uses a dictionary comprehension to construct a new dictionary called squares. The dictionary comprehension {x: x\*\*2 for x in range(10)} generates a dictionary with keys taken from the range 0 to 10 and values that are the square of each key. The print statement then outputs the resulting dictionary, which is {0: 0, 1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81}

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

Here is the code in Python to create a set called odd using a set comprehension:

odd = {x for x in range(10) if x % 2 != 0}

print(odd)

This code uses a set comprehension to construct a new set called odd. The set comprehension {x for x in range(10) if x % 2 != 0} generates a set of values x taken from the range 0 to 10 that satisfy the condition x % 2 != 0, i.e. it generates a set of odd numbers. The print statement then outputs the resulting set of odd numbers, which is {1, 3, 5, 7, 9}.

**7. 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.**

Here is the code in Python to use a generator comprehension to return the string 'got ' followed by each number in the range 0 to 10, and then iterate through the generator using a for loop:

generator = ('got ' + str(x) for x in range(10))

for item in generator:

print(item)

This code uses a generator comprehension ('got ' + str(x) for x in range(10)) to generate a generator that yields the string 'got ' followed by each number in the range 0 to 10 as a string. The for loop then iterates through the generator and prints each item.

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

def good():

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

print(good())

This code defines a function good that simply returns the list ['Harry', 'Ron', 'Hermione']. When the function is called with good(), the list is returned, and the print statement outputs the list, which is ['Harry', 'Ron', 'Hermione'].

**9. 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.**

Here is the code in Python to define a generator function get\_odds that returns the odd numbers from the range 0 to 10:

def get\_odds():

for x in range(10):

if x % 2 != 0:

yield x

for i, x in enumerate(get\_odds()):

if i == 2:

print("The third value returned is:", x)

break

This code defines a generator function get\_odds that yields the odd numbers from the range 0 to 10. The for loop for x in range(10): iterates through the range 0 to 10, and the if statement if x % 2 != 0: checks whether the current number x is odd. If x is odd, the generator function yields the value of x.

The second for loop for i, x in enumerate(get\_odds()): iterates through the generator function get\_odds and uses the enumerate function to keep track of the current index i of the iteration. The if statement if i == 2: checks whether the current index i is equal to 2. If it is, the code prints the message "The third value returned is:" followed by the value x of the third iteration, and the break statement terminates the loop. The output of this code will be The third value returned is: 5.

**10. 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'.**

Here is the code in Python to define an exception called OopsException and raise this exception to see what happens, followed by the code to catch the exception and print 'Caught an oops':

class OopsException(Exception):

pass

try:

raise OopsException

except OopsException:

print("Caught an oops")

This code defines a custom exception class OopsException that inherits from the built-in Exception class. In the try block, the code raises the OopsException with the raise OopsException statement.

The except block catches the OopsException and runs the code inside it, which is print("Caught an oops"). This will result in the output Caught an oops.

**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'].**

Here is the code in Python to use the zip function to make a dictionary called movies that pairs the lists titles = ['Creature of Habit', 'Crewel Fate'] and plots = ['A nun turns into a monster', 'A haunted yarn shop']:

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

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

movies = dict(zip(titles, plots))

print(movies)

This code first defines the two lists titles and plots. Then, the code uses the zip function to pair the elements of the two lists, which creates a sequence of tuples where each tuple consists of an element from each list. The dict constructor is then used to convert this sequence of tuples into a dictionary, which is stored in the variable movies.

The print statement at the end outputs the dictionary movies, which will be {'Creature of Habit': 'A nun turns into a monster', 'Crewel Fate': 'A haunted yarn shop'}