1. Calculate the following using Lambda calculus:

a. T AND F b. 3 * 4

```
true = lambda x: lambda y: x
false = lambda x: lambda y: y
result = (lambda x: lambda y: x)(true)(false)
if result(True)(False):
  print("T")
else:
  print("F")
```

```
T
```

```
(lambda x: (lambda y: x * y)(4))(3)
```

```
12
```

2. Lambda functions a. Write a lambda function to convert measurements from meters to feet. b. Write a lambda function in Python to implement the following lambda expression: (λλλλ. λλλλ. (ff + mm)aa)(λλλλ. xx2)(bb) Note: You need to write a nested lambda function for implementing f+m where f takes the square function (which takes argument x) passed as a parameter. The above

```
meters_to_feet = lambda meters: meters * 3.28084
meters = float(input("Enter a measurement in meters: "))
feet = meters_to_feet(meters)
# Display the result
print("{:.2f} meters is equivalent to {:.2f} feet.".format(meters, feet))
```

```
Enter a measurement in meters: 20
20.00 meters is equivalent to 65.62 feet.
```

```
# Define the lambda function
lambda_func = lambda a, b: (lambda f, m: (f(2) + m) * a)((lambda x: x * b), b)
# Prompt the user to enter two numbers
a = float(input("Enter the value of a: "))
b = float(input("Enter the value of b: "))
# Use the lambda function to calculate the result
result = lambda_func(a, b)
# Display the result
print("The result of the lambda expression is:", result)
```

```
Enter the value of a: 2
Enter the value of b: 3
The result of the lambda expression is: 18.0
```

3. Passing and returning a function as an argument Define a function ‘square’ for squaring a number. Define a function named ‘twice’ that takes a function f as an argument and returns f(f(x)). Using ‘twice’ and ‘square’ create a function ‘quad’ that takes n as an argument and returns n4. ‘quad’ should not be defined explicitly. It should only be created as a variable which is then assigned a function.

```
def square(x):
  return x**2
def twice(f):
  def g(x):
    return f(f(x))
  return g
quad = twice(square)
print(quad(2))
print(quad(3))
print(quad(4))
```

```
16
81
256
```

4. Closure A Closure is a function object that remembers values in enclosing scopes even if they are not present in memory. We have a closure in Python when a nested function references a value in its enclosing scope. a. Study the following program by executing it:

b. In a lottery system, random number is chosen by retrieving the number from a random index from a list of random numbers. Write a program to choose a random number in this way. You must use nested functions – the inner function chooses a number from a random index and the outer function generates a random list of numbers. The outer function takes n as a parameter where is the maximum number that can be put in the random list. (Your code should be similar to the program in 5a)

def multiplier_of(n): def multiplier(number): return number*n return multiplier

```
def multiplier_of(n):
  def multiplier(number):
    return number*n
  return multiplier
```

```
import random
def generate_random_list(n):
  def get_random_index(lst):
    return random.randint(0, len(lst)-1)
    random_list = [random.randint(1, n) for _ in range(n)]
    random_index = get_random_index(random_list)
    random_number = random_list[random_index]
    print("Random list:", random_list)
```

```
    print("Random index:", random_index)
    print("Random number:", random_number)
generate_random_list(10)
```

6. Map A secret message needs to be sent. Use the map function to encrypt the message using Caesar cipher.

```
def caesar_cipher(message, shift):
  encrypted = map(lambda char: chr((ord(char) - 97 + shift) % 26 + 97) if char.isal
  return ''.join(encrypted)
message = "secret message"
shift = 3
encrypted_message = caesar_cipher(message, shift)
print("Original message:", message)
print("Encrypted message:", encrypted_message)
```

```
Original message: secret message
Encrypted message: vhfuhw phvvdjh
```

7. Reduce Given runs scored by 2 players in a series of matches, write a Python program using reduce function to find who is the better player of the two in terms of maintaining consistency. (You need to find SD).

```
from functools import reduce
player1_runs = [56, 23, 45, 67, 34, 78, 56, 90, 87, 43]
player2_runs = [67, 89, 23, 56, 78, 45, 90, 34, 56, 87]
def calculate_sd(runs):
  mean = reduce(lambda x, y: x + y, runs) / len(runs)
  variance = reduce(lambda x, y: x + y, map(lambda x: (x - mean) ** 2, runs)) / len
  return variance ** 0.5
player1_sd = calculate_sd(player1_runs)
player2_sd = calculate_sd(player2_runs)
print("Player 1 SD:", player1_sd)
print("Player 2 SD:", player2_sd)
if player1_sd < player2_sd:
  print("Player 1 is the better player in terms of maintaining consistency.")
elif player2_sd < player1_sd:
  print("Player 2 is the better player in terms of maintaining consistency.")
else:
  print("Both players have the same level of consistency.")
```

```
Player 1 SD: 21.328150412072773
Player 2 SD: 22.544400635190993
Player 1 is the better player in terms of maintaining consistency.
```

8. Filter The marks scored by a class of students in 5 different subjects are stored in a list of lists. Using the filter function, write a program to find the students who failed in one or more subjects.

```
marks = [ [56, 34, 67, 78, 45],
[89, 78, 56, 23, 90],
[45, 56, 67, 34, 23],
[90, 56, 45, 78, 90],
[78, 90, 34, 56, 67],
[56, 67, 45, 23, 56],
[41, 45, 56, 90, 80] ]
failed_students = list(filter(lambda student: any(mark < 40 for mark in student), m
print("Students who failed in one or more subjects:")
for student in failed_students:
  print(student)
```

```
Students who failed in one or more subjects:
[56, 34, 67, 78, 45]
[89, 78, 56, 23, 90]
[45, 56, 67, 34, 23]
[78, 90, 34, 56, 67]
[56, 67, 45, 23, 56]
```

9. Map+reduce+filter Given two trending topics and a bunch of tweets, write a Python program to count the number of tweets that contain each topic. You need to do this by putting together map(), reduce() and filter() functions.

```
tweets = [
"Just saw a new movie and it was great!",
"I love going to the beach on sunny days",
"The new restaurant in town is amazing",
"Can't believe how much I'm enjoying this book",
"The weather today is terrible",
"This game is so boring",
"The movie I saw last night was terrible",
"I'm addicted to this new TV show"
]
topic1 = "movie"
topic2 = "restaurant"
tweets = list(map(lambda tweet: tweet.lower(), tweets))
tweets_filtered = list(filter(lambda tweet: topic1 in tweet or topic2 in tweet, twe
count_topic1 = reduce(lambda count, tweet: count + 1 if topic1 in tweet else count,
count_topic2 = reduce(lambda count, tweet: count + 1 if topic2 in tweet else count,
print(f"Number of tweets containing {topic1}: {count_topic1}")
print(f"Number of tweets containing {topic2}: {count_topic2}")
```

```
Number of tweets containing movie: 2
Number of tweets containing restaurant: 1
```

10. Given the list of scores secured by the students as a multidimesnion list Extracting Toppers From a Student Record print the index of top scorer in each subject using function tools

```
from functools import partial
def get_top_scorer_index(subject_scores):
  """
  Returns the index of the top scorer in the given list of subject scores.
  """
  index, score = max(enumerate(subject_scores), key=lambda x: x[1])
  return index
# Example student record
student_record = [ [80, 90, 85, 75, 95],
[70, 85, 90, 80, 75],
[95, 90, 80, 85, 70],
[75, 80, 70, 90, 80] ]# Use `map()` to apply `get_top_scorer_index()` to each subje
top_scorer_indices = list(map(get_top_scorer_index, zip(*student_record)))
# Print the index of the top scorer in each subject
for i, index in enumerate(top_scorer_indices):
  print(f"Subject {i+1} top scorer index: {index}")
```

```
Subject 1 top scorer index: 2
Subject 2 top scorer index: 0
Subject 3 top scorer index: 1
Subject 4 top scorer index: 3
Subject 5 top scorer index: 0
```

11. Given the list scores secured by the batsmans over a certain matches, compute the average score of individual batsman, also the individual highest score of the batsman

```
scores = [ [50, 20, 30, 40, 60],
[30, 40, 10, 50, 20],
[80, 60, 70, 40, 50] ]
averages = []
highest_scores = []
for batsman_scores in scores:
  average_score = sum(batsman_scores) / len(batsman_scores)
  highest_score = max(batsman_scores)
  averages.append(average_score)
  highest_scores.append(highest_score)
for i, (average_score, highest_score) in enumerate(zip(averages, highest_scores)):
  print(f"Batsman {i+1}: Average score = {average_score}, Highest score = {highest_
```

```
Batsman 1: Average score = 40.0, Highest score = 60
Batsman 2: Average score = 30.0, Highest score = 50
Batsman 3: Average score = 60.0, Highest score = 80
```

12. Calculate the number of grains of wheat on a chessboard given that the number on each square doubles. There once was a wise servant who saved the life of a prince. The king promised to pay whatever the servant could dream up. Knowing that the king loved chess, the servant told the king he would like to have grains of wheat. One grain on the first square of a chess board, with the number of grains doubling on each successive square. There are 64 squares on a chessboard (where square 1 has one grain, square 2 has two grains, and

so on). Write code that shows: how many grains were on a given square, and the total number of grains on the chessboard

```
square = int(input("Enter the square number: "))
grains_on_square = 2 ** (square - 1)
print(f"The number of grains on square {square} is {grains_on_square}")
total_grains = sum([2 ** i for i in range(64)])
print(f"The total number of grains on the chessboard is {total_grains}")
```

```
Enter the square number: 5
The number of grains on square 5 is 16
The total number of grains on the chessboard is 18446744073709551615
```