1. What is the relationship between def statements and lambda expressions ?

Ans:- A Lambda Function in Python programming is **an anonymous function or a function having no name**. ... In Python, lambda expressions (or lambda forms) are utilized to construct anonymous functions. To do so, you will use the lambda keyword (just as you use def to define normal functions).

I**def is the keyword for defining a function**. The function name is followed by parameter(s) in (). The colon : signals the start of the function body, which is marked by indentation. Inside the function body, the return statement determines the value to be returned.

1. What is the benefit of lambda?

Ans:- You can build serverless backends using AWS Lambda to handle web, mobile, Internet of Things (IoT), and 3rd party API requests. Take advantage of Lambda's **consistent performance controls**, such as multiple memory configurations and Provisioned Concurrency, for building latency-sensitive applications at any scale.

AWS Lambda allows **you to integrate your application server needs with mass-mailing services such as SES**. Thus, you can consolidate more of the functionality your team requires to operate under one house. This not only reduces administrative costs but also makes your team's workflow more streamlined and efficient.

1. Compare and contrast the maps, filter, and minimize the amount of data.

Ans:- map creates a new array by transforming every element in an array, individually. filter creates a new array by removing elements that don't belong. reduce , on the other hand, **takes all of the elements in an array and reduces them into a single value**. Just like map and filter , reduce is defined on Array.

map(), filter() and reduce()

1:map()

def starts with A(s):

return s[0]==”A”

fruit = [“Apple”,”Banana”,”Pear”,”Apricot”,”Orange”]

map\_object= map(starts with A,fruit)

print(list(map\_object))

2:filter()

def starts with A(s):

return s[0]==”A”

fruit = [“Apple”,”Banana”,”Pear”,”Apricot”,”Orange”]

filter\_object= filter(starts with A,fruit)

print(list(filter\_object))

3:reduce()

from functools import reduce

def add(x, y):

return x + y

list = [2, 4, 7, 3]

print(reduce(add, list))

4.What exactly are feature annotations and how do you use them?

Ans:- Each text annotation feature has **symbology including font, size, color, and any other text symbol property**. Annotation is typically text, but it can also include graphic shapes—for example, boxes or arrows—that require other types of symbology.

Feature-linked annotation is **a special type of geodatabase annotation that is directly linked to features**. Feature-linked annotation reflects the current state of features in the geodatabase: it is automatically updated when features are moved, edited, or deleted.

1. What exactly are recursive functions, and how do you use them?

Ans:- Recursion is a programming term that means **calling a function from itself**. Recursive functions can be used to solve tasks in elegant ways. When a function calls itself, that's called a recursion step. ... A recursively-defined data structure is a data structure that can be defined using itself.

So the main reason we use recursion is **to simplify (not optimize)** an algorithm into terms easily understood by most people. A classic example is the binary search. The algorithm for binary search in plain English: Start with a sorted collection of data (like a telephone book). People use recursion **only when it is very complex to write iterative code**

1. What are some general coding role design guidelines?

Ans:-

* Safe: It can be used without causing harm.
* Secure: It can't be hacked.
* Reliable: It functions as it should, every time.
* Testable: It can be tested at the code level.
* Maintainable: It can be maintained, even as your codebase grows.
* Portable: It works the same in every environment.

**PEP 8 : Coding Style guide in Python**

* Use 4-space indentation and no tabs. ...
* Use docstrings : There are both single and multi-line docstrings that can be used in Python. ...
* Wrap lines so that they don't exceed 79 characters : The Python standard library is conservative and requires limiting lines to 79 characters.

1. Describe three or more ways in which functions can send results to a caller.

Ans:- function return a value, use the return statement

Call By Value. Call By Reference. While calling a function, we **pass values of variables to it**. Such functions are known as “Call By Values”. While calling a function, instead of passing the values of variables, we pass address of variables(location of variables) to the function known as “Call By References.

In Call by value, a copy of the variable is passed whereas in Call by reference, **a variable itself is passed**. In Call by value, actual and formal arguments will be created in different memory locations whereas in Call by reference, actual and formal arguments will be created in the same memory location.

**Python always uses pass-by-reference values**. There isn't any exception. Any variable assignment means copying the reference value.

Python is **“pass-by-object-reference**”, of which it is often said: “Object references are passed by value.”

In Python, arguments are always passed by value, and **return values are always returned by value**. However, the value being returned (or passed) is a reference to a potentially shared, potentially mutable object.

1)\*args

def my\_function(\*kids):  
  print("The youngest child is " + kids[2])  
  
my\_function("Emil", "Tobias", "Linus")

outout:-The youngest child is Linus

2)Keyword Arguments

def my\_function(child3, child2, child1):  
  print("The youngest child is " + child3)  
  
my\_function(child1 = "Emil", child2 = "Tobias", child3 = "Linus")

Output:-The youngest child is Linus

3)Arbitrary keyword Arguments(\*\*kwargs)

def my\_function(\*\*kid):  
  print("His last name is " + kid["lname"])  
  
my\_function(fname = "Tobias", lname = "Refsnes")

output:-His last name is Refsnes

4) def my\_function(x):  
  **return 5 \* x**  
print(my\_function(3))  
output:-15

5) list as argument

def my\_function(food):  
  for x in food:  
    print(x)  
  
fruits = ["apple", "banana", "cherry"]  
  
my\_function(fruits)

output:- apple,

  banana,

cherry

6)Recursion

def tri\_recursion(k):  
  if(k > 0):  
    result = k + tri\_recursion(k - 1)  
    print(result)  
  else:  
    result = 0  
  return result  
  
print("\n\nRecursion Example Results")  
tri\_recursion(6)

output:- Recursion Example Results

1

3

6

10

15

21