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

**Answer:**

**Like def, the lambda creates a function to be called later. But it returns the function instead of assigning it to a name. This is why lambdas are sometimes known as anonymous functions.**

**def sum(a,b):**

**return a+b**

**lamb = lambda a,b : a + b**

**a=10**

**b=20**

**print(sum(a,b))**

**print(lamb(a,b))**

2. What is the benefit of lambda?

**Answer:**

1. **The code is simple and clear.**
2. **No additional variables are added.**

3. Compare and contrast map, filter, and reduce.

**Answer:**

**The map(), filter() and reduce() functions bring a bit of functional programming to Python. All three of these are convenience functions that can be replaced with List Comprehensions or loops.**

**map()**

**The map() function iterates through all items in the given iterable and executes the function we passed as an argument on each of them.**

**e.g : map(function, iterable(s))**

**filter()**

**filter() forms a new list that contains only elements that satisfy a certain condition, i.e. the function we passed returns True.**

**e.g: filter(function, iterable(s))**

**reduce()**

**reduce() works differently than map() and filter(). It does not return a new list based on the function and iterable we've passed. Instead, it returns a single value.**

**e.g:**

**from functools import reduce**

**def add(x, y):**

**return x + y**

**list = [2, 4, 7, 3]**

**print(reduce(add, list)) 🡪16 #add all the list elements and returns the single value**

4. What are function annotations, and how are they used?

**Answer:**

**Function annotations adds a feature that allows you to add arbitrary metadata to function parameters and return value. Function annotations are completely optional both for parameters and return value.**

**The annotations for parameters take the following form −**

**e.g,**

**def example(x: expression, y: expression = 1):**

**….function body….**

5. What are recursive functions, and how are they used?

**Answer:**

**A recursive function is a function that calls itself during its execution. Function calling itself is know as recursive function.**

**e.g**

**def factorial(num):**

**if num == 1:**

**return 1**

**else:**

**return (num \* factorial(num-1))**

**x = int(input("please enter a number"))**

**print("The factorial of", x, "is", factorial(x))**

6. What are some general design guidelines for coding functions?

**Answer:**

**Some of the general design guidelines are as below:**

* **Limited use of globals**
* **Naming conventions for local variables, global variables, constants and functions**
* **Indentation**
* **Exception handling conventions.**
* **Code should be well documented.**
* **Code should be understandable.**

7. Name three or more ways that functions can communicate results to a caller.

**Answer:**

1. **By using return single value.**
2. **By using return multiple values.**
3. **By returning a function.**