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

|  |  |
| --- | --- |
| def defined functions | lambda functions |
| Easy to interpret | Interpretation might be tricky |
| Can consists of any number of execution statements inside the function definition | The limited operation can be performed using lambda functions |
| To return an object from the function, return should be explicitly defined | No need of using the return statement |
| Execution time is relatively slower for the same operation performed using lambda functions | Execution time of the program is fast for the same operation |
| Defined using the keyword def and holds a function name in the local namespace | Defined using the keyword lambda and does not compulsorily hold a function name in the local namespace |

2. What is the benefit of lambda?

|  |
| --- |
| lambda functions |
| Interpretation might be tricky |
| The limited operation can be performed using lambda functions |
| No need of using the return statement |
| Execution time of the program is fast for the same operation |
| Defined using the keyword lambda and does not compulsorily hold a function name in the local namespace |

3. Compare and contrast map, filter, and reduce.  
Ans:   
Function annotations are completely optional both for parameters and return value.  
Function annotations provide a way of associating various parts of a function with arbitrary python expressions at compile time.

Map: - map() takes a function object and an iterable and creates a new list. The map() function iterates through all items in the given iterable and executes the function we passed as an argument on each of them.  
Filter:- Similar to map(), filter() takes a function object and an iterable and creates a new list. As the name suggests, filter() forms a new list that contains only elements that satisfy a certain condition, i.e. the function we passed returns True.  
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.

4. What are function annotations, and how are they used?  
Function annotations are completely optional both for parameters and return value.  
Function annotations provide a way of associating various parts of a function with arbitrary python expressions at compile time.

5. What are recursive functions, and how are they used?  
Ans: Python also accepts function recursion, which means a defined function can call itself.  
The developer should be very careful with recursion as it can be quite easy to slip into writing a function which never terminates, or one that uses excess amounts of memory or processor power. However, when written correctly recursion can be a very efficient and mathematically-elegant approach to programming.

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

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.

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