**Python Assignment- 24**

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

**Ans**: The def keyword is used to define normal functions, while the lambda keyword is used to define anonymous functions. They are, however, limited to a single line of expression. They, like regular functions, can accept several parameters.

2. What is the benefit of lambda?

**Ans:** Lambda helps you use a function only once, and hence, avoids cluttering up the code with function definitions. In short, Python's lambda keyword lets you define a function in a single line of code and use it immediately.

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

**Ans:**  map applies as a transformation to an element.

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

filter accumulates only elements matching a condition.

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

reduce accumulates all elements to a single value, by using immutable values

reduce() works by calling the function we passed for the first two items in the sequence. The result returned by the function is used in another call to function alongside with the next (third in this case), element Syntax : reduce(function, sequence[, initial])

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

**Ans:** Function annotations are some random expressions which are written with the functions, and they are evaluated at compile time. They do not exist at run time, and there is no meaning of these expressions to python. They are used and interpreted by a third party or external python libraries.

* Syntax :
  + def func(a: 'int') -> 'int':
  + pass
* Annotations for simple parameters:
  + def func(x: 'float'=10.8, y: 'argument2'):
    - In the above code the argument, ‘x’ of the function func, has been annotated to float data type and the argument ‘y’ has a string-based annotation. The argument can also be assigned to a default value using a ‘=’ symbol followed by the default value. These default values are optional to the code.
* Annotations for return values:
  + def func(a: expression) -> 'int':
    - The annotations for the return value is written after the ‘->’ symbol.

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

**Ans**: A recursive function is a function in code that refers to itself for execution. Recursive functions can be simple or elaborate. They allow for more efficient code writing, for instance, in the listing or compiling of sets of numbers, strings or other variables through a single reiterated process.

This means that the function will continue to call itself and repeat its behavior until some condition is met to return a result.

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

**Ans:** Following are the some general design guidelines for coding functions.

1. Limited use of globals:  
   These rules tell about which types of data that can be declared global and the data that can’t be.
2. Standard headers for different modules:  
   For better understanding and maintenance of the code, the header of different modules should follow some standard format and information. The header format must contain below things that is being used in various companies:
   * Name of the module
   * Date of module creation
   * Author of the module
   * Modification history
   * Synopsis of the module about what the module does
   * Different functions supported in the module along with their input output parameters
   * Global variables accessed or modified by the module
3. Naming conventions for local variables, global variables, constants and functions:  
   Some of the naming conventions are given below:
   * Meaningful and understandable variables name helps anyone to understand the reason of using it.
   * Local variables should be named using camel case lettering starting with small letter (e.g. localData) whereas Global variables names should start with a capital letter (e.g. GlobalData). Constant names should be formed using capital letters only (e.g. CONSDATA).
   * It is better to avoid the use of digits in variable names.
   * The names of the function should be written in camel case starting with small letters.
   * The name of the function must describe the reason of using the function clearly and briefly.
4. Indentation:  
   Proper indentation is very important to increase the readability of the code. For making the code readable, programmers should use White spaces properly. Some of the spacing conventions are given below:
   * There must be a space after giving a comma between two function arguments.
   * Each nested block should be properly indented and spaced.
   * Proper Indentation should be there at the beginning and at the end of each block in the program.
   * All braces should start from a new line and the code following the end of braces also start from a new line.
5. Error return values and exception handling conventions:  
   All functions that encountering an error condition should either return a 0 or 1 for simplifying the debugging.

On the other hand, Coding guidelines give some general suggestions regarding the coding style that to be followed for the betterment of understandability and readability of the code. Some of the coding guidelines are given below :

1. Avoid using a coding style that is too difficult to understand:  
   Code should be easily understandable. The complex code makes maintenance and debugging difficult and expensive.
2. Avoid using an identifier for multiple purposes:  
   Each variable should be given a descriptive and meaningful name indicating the reason behind using it. This is not possible if an identifier is used for multiple purposes and thus it can lead to confusion to the reader. Moreover, it leads to more difficulty during future enhancements.
3. Code should be well documented:  
   The code should be properly commented for understanding easily. Comments regarding the statements increase the understandability of the code.
4. Length of functions should not be very large:  
   Lengthy functions are very difficult to understand. That’s why functions should be small enough to carry out small work and lengthy functions should be broken into small ones for completing small tasks.
5. Try not to use GOTO statement:  
   GOTO statement makes the program unstructured, thus it reduces the understandability of the program and also debugging becomes difficult.
6. Use Python’s default UTF-8 or ASCII encodings and not any fancy encodings.

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

**Ans:**

1. Function can return single value
2. Can return multiple values, tuple
3. can return list,dictionary
4. can return function object
5. can return class object