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

**Answer :**

1. Keyword def that marks the start of the function header. A function name to uniquely identify the function. Function naming follows the same rules of writing identifiers in Python.
2. Lambdas are one-line methods without a name or we can say anonymous functions. They work practically the same as any other method in Python. Lambdas differ from normal Python methods because they can have only one expression, can't contain any statements and their return type is a function object

2. What is the benefit of lambda?

**Answer :**

1. Fewer Lines of Code.
2. Lambda functions are inline functions and thus execute comparatively faster.
3. Many times lambda functions make code much more readable by avoiding the logical jumps caused by function calls

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

**Answer :**

1. Map - It 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))

1. Filter - It 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))

1. Reduce - It 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?

**Answer :**

* Function annotations are arbitrary python expressions that are associated with various part of functions. These expressions are evaluated at compile time and have no life in python’s runtime environment. Python does not attach any meaning to these annotations. They take life when interpreted by third party libraries, for example, mypy.
* They are like the optional parameters that follow the parameter name.

Syntax :

(i) **Annotations for simple parameters :** The argument name is followed by ‘:’ which is then followed by the expression.

Syntax => def foobar(a: expression, b: expression = 5):

(ii) **Annotations for excess parameters :** Excess parameters for e.g. \*args and \*\*kwargs, allow arbitrary number of arguments to be passed in a function call.

Syntax => def foobar(\*args: expression, \*\*kwargs: expression = 5):

(iii) **Annotations for nested parameters :** Nested parameters are useful feature of python 2x where a tuple is passed in a function call and automatic unpacking takes place. This feature is removed in python 3x and manual unpacking should be done. Annotation is done after the variable and not after the tuple as shown below.

Syntax => def foobar((a: expression, b: expression), (c: expression, d: expression)):

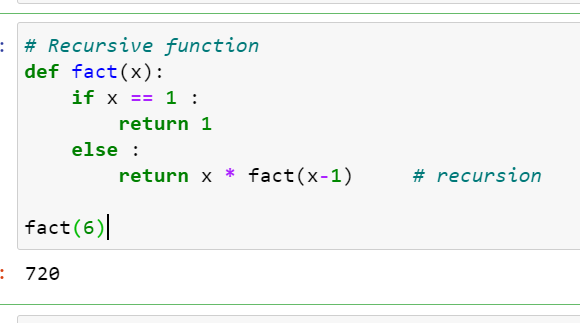
(iv) Annotations for return type : Annotating return type is slightly different from annotating function arguments. The ‘->’ is followed by expression which is further followed by ‘:’.

Syntax => def foobar(a: expression) -> expression:

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

**Answer :**

Recursion is a common mathematical and programming concept. It means that a function calls itself. This has the benefit of meaning that you can loop through data to reach a result.



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

**Answer :**

1. Use 4-space indentation and no tabs.
2. Use docstrings
3. Wrap linethat they don’t exceed 79 characters
4. Use of regular and updated comments are valuable to both the coders and users
5. Use of trailing commas : in case of tuple -> ('good',)
6. Use Python’s default UTF-8 or ASCII encodings and not any fancy encodings
7. Naming Conventions 8.Characters that should not be used for identifiers : ‘l’ (lowercase letter el), ‘O’ (uppercase letter oh), ‘I’ (uppercase letter eye) as single character variable names as these are similar to the numerals one and zero.
8. Don’t use non-ASCII characters in identifiers
9. Name your classes and functions consistently
10. While naming of function of methods always use self for the first argument

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

**Answer :**

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