Python Basics Assignment 24

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

Ans: def statement is used to create a normal function whereas lambda expressions are used to create Anonymous functionswhich can be assigned to a variable and can be called using the variable later in function.

Lambda's body is a single expression and not a block of statements like def statement. The lambda expression's body is similar to what we wouldd put in a def body's return statement. We simply type the result as an expression instead of explicitly returning it. Because it is limited to an expression, a lambda is less general than a def statement.

Example of Lambda Function is shown below:

```
x = lambda a, b, c : a + b + c
print(x(5, 6, 2))
```

13

2. What is the benefit of lambda?

Ans: The following are some of the benefits of lambda expressions:

1. It can be used to create Nameless/Anonymous functions inside some complex functions if we are planning to use it only once.
2. Moderate to small functions can be created in a single line
3. Fuctions created using lambda expressions can be assigned to a variable and can be used by simply calling the variable

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

Ans: The differences between map, filter and reduce are:

1. map(): The map() function is a type of higher-order. This function takes another function as a parameter along with a sequence of iterables and returns an output after applying the function to each iterable present in the sequence.
2. filter(): The filter() function is used to create an output list consisting of values for which the function returns true.
3. reduce(): The reduce() function, as the name describes, applies a given function to the iterables and returns a single value

```
from functools import reduce
# map function
print('Map ->',list(map(lambda x:x+x, [1,2,3,8])))
# fitler function
print('Filter ->',list(filter(lambda x:x%2 !=0, [1,2,3,9])))
# reduce function
print('Reduce ->',reduce(lambda x,y:x+y, [1,2,3,4,5,7]))
```

Map -> [2, 4, 6, 16]
Filter -> [1, 3, 9]
Reduce -> 22

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

Ans: Function annotations provide a way of associating various parts of a function with arbitrary python expressions at compile time.
Annotations of simple parameters def func(x: expression, y: expression = 20):
Whereas the annotations for excess parameters are as − def func (**args: expression, **kwargs: expression):

Purpose of Function Annotations:

1. Python supports dynamic typing and hence no module is provided for type checking.
2. String based annotations can be used by the libraries to provide better help messages at compile time regarding the functionalities of various methods, classes and modules.

Example of Annotations:

1. Annotations for simple parameters : def foo(a: expression, b: expression = 5):
2. Annotations for excess parameters : def foo(*args: expression*, kwargs: expression):
3. Annotations for nested parameters : def foo((a: expression, b: expression), (c: expression, d: expression)):

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

Ans: A recursive function is a function that calls itself during its execution. The process may repeat several times, outputting the result and the end of each iteration. An example is mentioned below:

```python
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(9)
```

Recursion Example Results
1
3
6
10
15
21
28
36
45

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

Ans: Some of the general design guidelines for coding functions are:

1. Always use a docstring to explain the functionality of the function
2. Avoid using or limit use of global variables
3. Proper Indentation to increase the code readability
4. Try to follow a proper naming convention for function names (pascalCase or camelCase) and stick with the same convention throughout the application.
5. Avoid using digits while choosing a variable name
6. Try to use a name for the function which conveys the purpose of the function
7. Local variables should be named using camelCase format (ex: localVariable) whereas Global variables names should be using PascalCase (ex:GlobalVariable).
8. Constant should be represented in allcaps (ex:CONSTANT).

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

Ans: Some of the ways in which a function can communicate with the calling function is:

1. print
2. return
3. yield