Assignment 24 Soluitons

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 functions` which 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.

2. What is the benefit of lambda?

ANS:

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

In [1]:

```python
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 annotation is the standard way to access the metadata with the arguments and the return value of the function. These are nothing but some random and optional Python expressions that get allied to different parts of the function. They get evaluated only during the compile-time and have no significance during the run-time of the code. They do not have any significance or meaning associated with them until accessed by some third-party libraries. They are used to type check the functions by declaring the type of the parameters and the return value for the functions. The string-based annotations help us to improve the help messages.

```
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.
```

In [2]:

```python
def fib(n:'float', b:'int')-> 'result':
    pass
print(fib.__annotations__)
```

```
{'n': 'float', 'b': 'int', 'return': 'result'}
```

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

ANS: A recursive function is a function that calls itself during its execution. This means that the function will continue to call itself and repeat its behavior until some condition is met to return a result

In [3]:

```python
def fact(x):
    if x == 1 :
        return 1
    else :
        return x * fact(x-1) # recurtion

fact(3)
```

Out[3]:

```
6
```

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

ANS:

```
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 sim
ilar to the numerals one and zero.
9. Don’t use non-ASCII characters in identifiers
10. Name your classes and functions consistently
11. 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. ¶

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
```