**Q1. What is the difference between a function and a method in Python?**

**Ans:** Function and Method both look similar as they perform in almost similar way, but the key difference is the concept of Class and its Object.

Functions:  Functions can be called only by its name, as it is defined independently.
Method: Method can't be called by its name only, we need to invoke the class by a reference of that class in which it is defined, method is defined within a class and hence they are dependent on that class.

**Q2. Explain the concept of function arguments and parameters in Python?**

**Ans:** Arguments are the values passed to the function when it's called. They provide the flexibility to pass different values to the function. Arguments can be mutable or immutable depending on their data type.Parameters are listed in the parentheses of a function's definition. They act as placeholders for the data the function will use. Parameters define the function's signature and the number and type of inputs it can accept

```
def rectangle(x, y):
   print("Rectangle have (Length:", x, ", Breadth:", y, ")")
   rectangle(y=10, x=5)
```

Here parameters are x and y Which is been mentioned at declaration of function where as at function call we provided values that are arguments.

**Q3. What are the different ways to define and call a function in Python?**
**Ans:**
1:  Simple Function Definition
```
def greet(name):
   print(f"Hello, {name}!")
greet("Pankaj")
```

2:  Lambda Function Or Anonymous Function

```
a = lambda name:
  print(f"Hello, {name}!")
a("Pankaj")
```

**Q4. What is the purpose of the `return` statement in a Python function?**

**Ans:** The return statement is used in a function, but not outside of a function. Using the return statement outside of a function will throw an error.

- A function can have multiple return statements, but only one will be executed depending on the condition.

- The return statement can return data of any type, including integers, floats, strings, lists, dictionaries, and even other functions.
- If a function doesn't have any return statement, then it returns None.
- You can use expressions in the return statement. The expression is evaluated and the result is returned

Q5. What are iterators in Python and how do they differ from iterables?
Ans:

| **Basic** | **Iterable** | **Iterator** |
|---|---|---|
| **What** | An object to iterate over is Iterable. | Iterators are defined as an object that counts iteration via an internal state variable. |
| **Relation** | Every iterator is iterable. | Not every iterable is an iterator. |
| **Method for Iterating** | An object which would generate an iterator when passed to in-built method iter(). | The next() is used for iterating. |
| **List** | A List is iterable. | A List is not an iterator. |

**Q6. Explain the concept of generators in Python and how they are defined.**

**Ans:** A generator function is a special type of function that returns an iterator object. Instead of using return to send back a single value, generator functions use yield to produce a series of results over time. This allows the function to generate values and pause its execution after each yield, maintaining its state between iterations.

```
def fun():
   yield 1
   yield 2
   yield 3
for val in fun():
   print(val)
```

Output:-

```
1
2
3
```

**Q7. What are the advantages of using generators over regular functions?**

**Ans**: Generators differ from standard functions in this it allow you to iterate through a sequence of values over time, instead of computing and returning a single result.
Standard Functions: Compute a value and return it once. They use the return statement.

```
def standard_function():
    return [1, 2, 3]
```

Generators: Use the yield statement to return values one at a time, allowing iteration over a sequence of values without storing the entire sequence in memory.

```
def generator_function():
    yield 1
    yield 2
    yield 3
```

**Q8. What is a lambda function in Python and when is it typically used?**

**Ans:** A lambda function is a small, anonymous function that's defined using
the lambda keyword. Lambda functions are typically used for short, simple operations that can be written in a single line. They are often used in functional programming contexts with higher-order functions like map, filter, and reduce.

**Q9. Explain the purpose and usage of the `map()` function in Python.**

**Ans:** The map() function takes two arguments: a function and an iterable. The function contention is the function that will be applied to every element of the iterable, and the iterable contention is the iterable that the function will be applied to. Here is the syntax of the map() function

**Q10. What is the difference between `map()`, `reduce()`, and `filter()` functions in Python?**
Ans:

- **Map()** : The map function takes a function and applies it to each element in an iterable, such as a list or tuple. The result is a new iterable containing the transformed values.
- Ex:

```
numbers = [1, 2, 3, 4, 5]
doubled_numbers = list(map(lambda x: x * 2, numbers))
print(doubled_numbers)

[2, 4, 6, 8, 10]
```

- **REDUCE:** The reduce() function takes a function and an iterable and applies the function to the first two elements, then to the result and the next element, and so on, until all elements have been processed. The result is a single value.
- EX:

```
from functools import reduce
numbers = [1, 2, 3, 4, 5]
total = reduce(lambda x, y: x + y, numbers)
print(total)

15
```

- Filter: The filter() function takes a function and an iterable and returns a new iterable containing only the elements for which the function returns True.

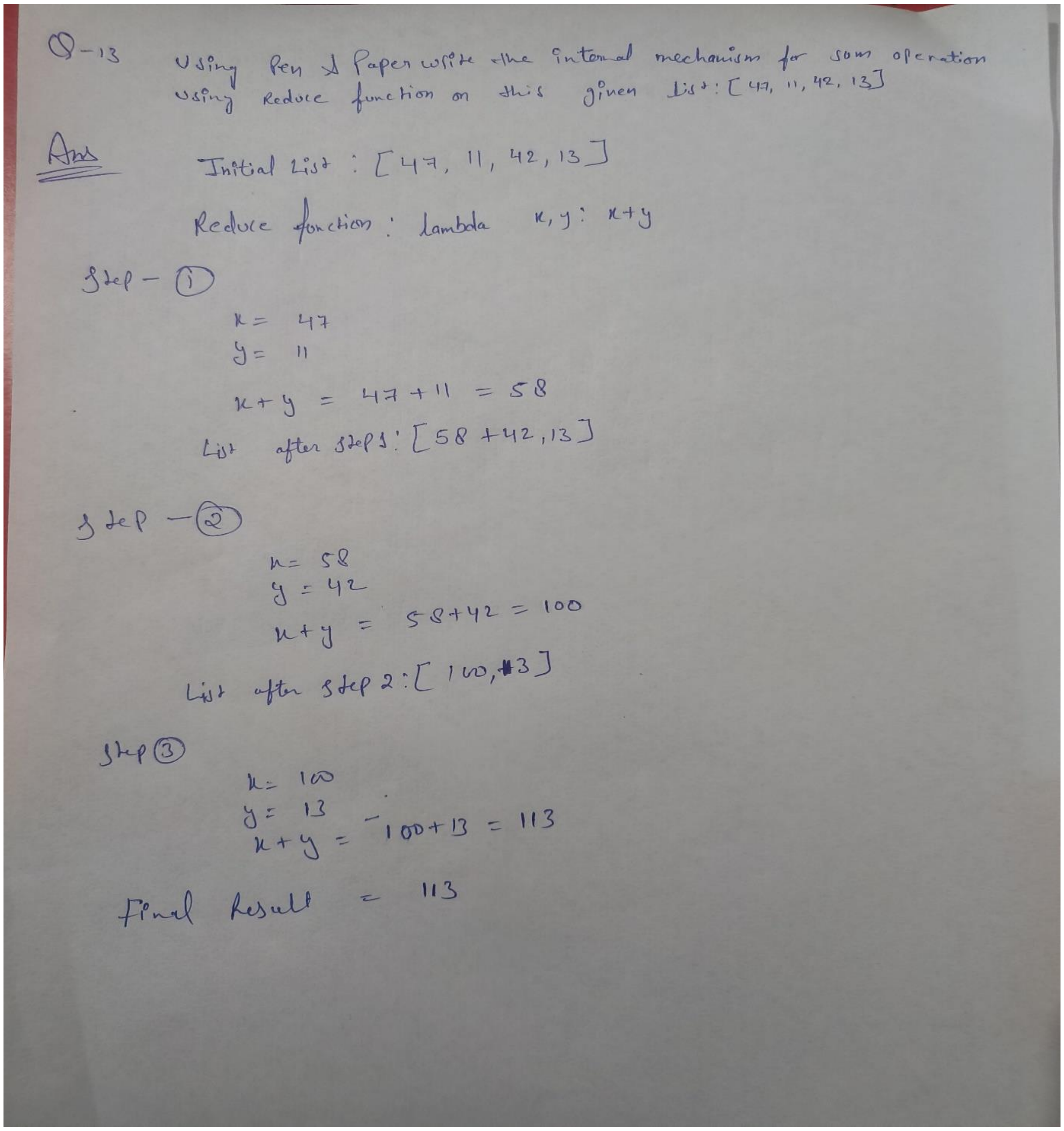
- Ex:

```
numbers = [1, 2, 3, 4, 5]
odds = list(filter(lambda x: x % 2 != 0, numbers))
print(odds)
```

```
[1, 3, 5]
```

Q11: Using pen & Paper write the internal mechanism for sum operation using  reduce function on this given list:[47,11,42,13].

Ans:



Q-13 Using Pen & Paper write the internal mechanism for sum operation using Reduce function on this given list: [47, 11, 42, 13]

Ans

Initial List : [47, 11, 42, 13]

Reduce function : lambda x, y : x+y

Step - ①

x = 47
y = 11

x+y = 47+11 = 58

List after step 1 : [58 +42, 13]

Step - ②

x = 58
y = 42
x+y = 58+42 = 100

List after step 2 : [100, 13]

Step ③

x = 100
y = 13
x+y = 100+13 = 113

Final Result = 113