1. What is a lambda function in Python, and how does it differ from a regular function?

A) In Python, a lambda function, also known as an anonymous function, is a small, one-line function that doesn't require a name. It is created using the lambda keyword and is typically used for simple, short-lived operations. The basic syntax of a lambda function is as follows:

lambda arguments: expression

Here, arguments refers to the parameters of the function, and expression is the computation that the function performs. The result of the expression is automatically returned when the lambda function is called.

Lambda functions have a few key differences compared to regular functions:

Syntax: Lambda functions are defined using a more compact syntax compared to regular functions. They are typically written as a single line of code.

Namelessness: Lambda functions are anonymous, meaning they don't have a name assigned to them. They are usually used where they are defined or passed directly as arguments to other functions.

Single Expression: Lambda functions are limited to a single expression, which is evaluated and returned automatically. They are not designed to contain complex statements or multiple lines of code.

Function Objects: Lambda functions create function objects, just like regular functions. However, lambda functions are usually used as throwaway functions for specific purposes, whereas regular functions are defined with a name and can be reused throughout a program.

Here's an example to demonstrate the usage of a lambda function:

```
# Regular function
def square(x):
    return x ** 2

print(square(5))  # Output: 25

# Lambda function
square_lambda = lambda x: x ** 2

print(square_lambda(5))  # Output: 25
```

In this example, both the regular function square and the lambda function square_lambda perform the same operation, which is squaring a given number. However, the lambda function is defined in a more concise manner without explicitly naming it.