**Lambda:**

* Lambda functions, also known as anonymous functions, are small, single-expression functions that don't require a def statement or a return statement.
* They are defined using the lambda keyword in Python.
* The general syntax of a lambda function is : input: return

Here's a breakdown of each part:

* **lambda**: This keyword is used to define a lambda function.
* **arguments**: These are the input arguments to the function. Like regular functions, lambda functions can take any number of arguments, separated by commas. However, they are limited to a single expression, so complex function signatures are not supported.
* **expression**: This is the single expression that the lambda function evaluates and returns. The result of this expression becomes the return value of the lambda function.

Lambda functions are often used in situations where a simple function is required for a short period and doesn't need to be named. They are commonly used in conjunction with functions that accept other functions as arguments, such as map(), filter(), and sorted(), or in situations where a small function is needed inline, such as with the key parameter of sorted().

Here's an example of a lambda function that squares a number:

square = lambda x: x \*\* 2

print(square(5)) # Output: 25

In this example, the lambda function **lambda x: x \*\* 2** takes one argument **x** and returns the square of **x**. The lambda function is assigned to the variable **square**, and we call **square(5)** to compute the square of 5, which results in 25.

Lambda functions can be very useful for writing concise and readable code, especially in situations where a full function definition would be overkill. However, they should be used judiciously, and complex logic is generally better expressed using a named function definition.