lambda

In Python, a `lambda` function, also known as an anonymous function, is a concise way to create small, unnamed functions. `lambda` functions are often used for short-lived operations where a full function definition is unnecessary. They are defined using the `lambda` keyword, followed by a list of parameters, a colon, and an expression. The result of the expression is implicitly returned from the `lambda` function.

Here's an explanation with examples:

Syntax of a `lambda` function:

```python
lambda arguments: expression
```

Example 1: Basic `lambda` function:

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

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

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

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

In this example, `square_lambda` is a `lambda` function that squares its input, equivalent to the regular `square` function.

Example 2: Using `lambda` with `map()`:

The `lambda` function is often used in combination with built-in functions like `map()`.

```python
numbers = [1, 2, 3, 4, 5]

# Using lambda with map to square each element in the list
squared_numbers = list(map(lambda x: x ** 2, numbers))
print(squared_numbers)  # Output: [1, 4, 9, 16, 25]
```

Here, the `lambda` function squares each element in the `numbers` list, and `map()` applies this function to each element of the list.

Example 3: Using `lambda` with `filter()`:

Similarly, `lambda` can be used with the `filter()` function.

```python
```

```
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9]

# Using lambda with filter to select even numbers
even_numbers = list(filter(lambda x: x % 2 == 0, numbers))
print(even_numbers)  # Output: [2, 4, 6, 8]
```

Here, the `lambda` function filters out the even numbers from the `numbers` list using `filter()`.

Example 4: Sorting with `lambda`:

`lambda` is commonly used with the `sorted()` function for custom sorting.

```python
students = [
    {'name': 'Alice', 'grade': 95},
    {'name': 'Bob', 'grade': 88},
    {'name': 'Charlie', 'grade': 92}
]

# Sorting students based on their grades using lambda
sorted_students = sorted(students, key=lambda x: x['grade'], reverse=True)
print(sorted_students)
# Output: [{'name': 'Alice', 'grade': 95}, {'name': 'Charlie', 'grade': 92}, {'name': 'Bob', 'grade': 88}]
```

Here, the `lambda` function extracts the 'grade' field for sorting the list of dictionaries.

Note:

While `lambda` functions are concise and useful for simple operations, they are limited compared to regular functions. `lambda` functions can only contain a single expression, and their use is generally limited to cases where a short function is needed for a specific task.

In many cases, especially for more complex operations or functions that will be reused, it's often clearer to use a regular named function. `lambda` functions are a tool for writing quick, one-time-use functions.