Lesson Plan

Lambda Functions

PW SKILLS

Topics to be covered:

1. What are Lambda Functions?
2. Working Process of Lambda Functions
3. Lambda Functions Examples
4. Analogy of Lambda Functions
5. Analogy with Lambda Functions Examples

1. What are Lambda Functions?

- A lambda function in Python is a concise way to create anonymous functions, also known as lambda expressions.
- Unlike regular functions defined using the def keyword, lambda functions are often used for short-term operations and are defined in a single line.

The basic syntax of a lambda function is:

```
lambda arguments: expression
```

Here, arguments are the input parameters, and expression is the operation to be performed using those parameters. Lambda functions can have any number of input parameters but can only have one expression.

2. Working Process of Lambda Function:

1. Lambda Function Creation:

- First, you create a lambda function using the `lambda` keyword.
- The basic syntax is: `lambda arguments: expression`.

Example:

```
add = lambda x, y: x + y
```

2. Arguments:

- Lambda functions can take any number of arguments (input parameters).
- In the example, `x` and `y` are the arguments.

3. Expression:

- First, you create a lambda function using the `lambda` keyword.
- The basic syntax is: `lambda arguments: expression`.

4. Function Object:

- He lambda function creation results in a function object.
- This object can be assigned to a variable, like `add` in our example.

5. Function Call:

- When you want to use the lambda function, you call it like any other function, providing values for its arguments.
- In our example, calling `add(3, 4)` means substituting `x` with `3` and `y` with `4` in the lambda expression.

6. Expression Execution:

- The lambda expression is then executed with the provided arguments.
- In our example, `3 + 4` is calculated, resulting in `7`.

7. Result:

- The result of the lambda expression becomes the result of the function call.
- In our example, `add(3, 4)` returns `7`.

The summarized process is:

1. Create Lambda Function:

```
add = lambda x, y: x + y
```

2. Call Lambda Function:

```
result = add(3, 4)
```

3. Lambda Expression Execution:

- `x` is replaced with `3`.
- `y` is replaced with `4`.
- `3 + 4` is calculated.
- Result (`7`) is returned.

Lambda functions are useful for short-term, specific tasks, where you don't need a full-fledged function with a name and extensive logic.

3. Lambda Functions Examples:

1. Addition of Two Numbers:

```
add = lambda x, y: x + y
result = add(3, 4)
print(result)  # Output: 7
```

2. Check for Even Numbers:

```
is_even = lambda x: x % 2 == 0
print(is_even(10))  # Output: True
print(is_even(15))  # Output: False
```

3. Calculate Square of a Number:

```
square = lambda x: x**2
result = square(5)
print(result)  # Output: 25
```

4. Sort List of Words by Length:

```
words = ["Python", "DataScience", "JS",
"MachineLearning"]
sorted_words = sorted(words, key=lambda word: len(word))
print(sorted_words)
# Output: ['JS', 'Python', 'DataScience',
'MachineLearning']
```

5. Generate Fibonacci Sequence:

```
fibonacci = lambda n: n if n ⩽ 1 else fibonacci(n-1) +
fibonacci(n-2)
sequence = [fibonacci(i) for i in range(10)]
print(sequence)

# Output: [0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
```

4. Analogy of Lambda Functions: