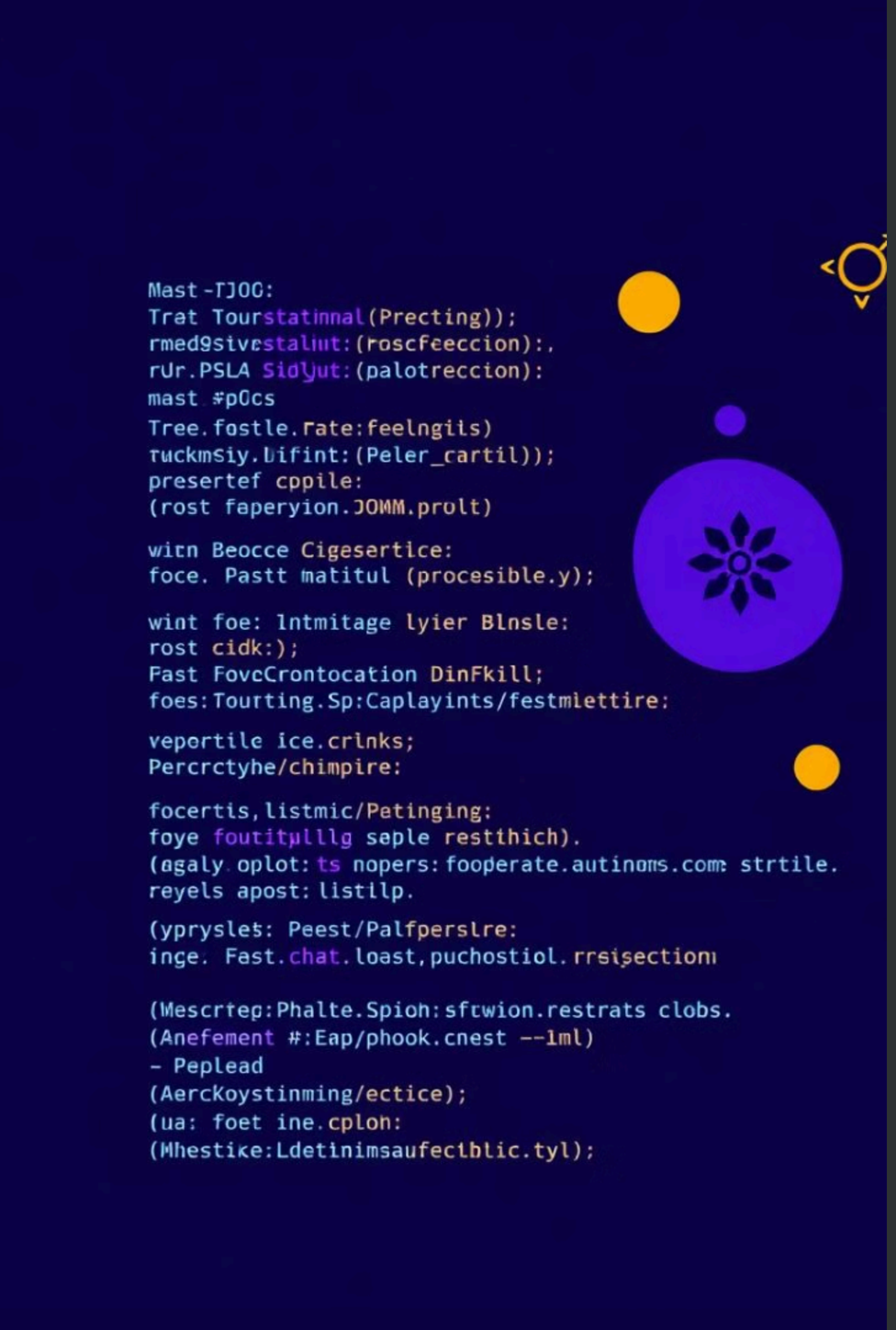
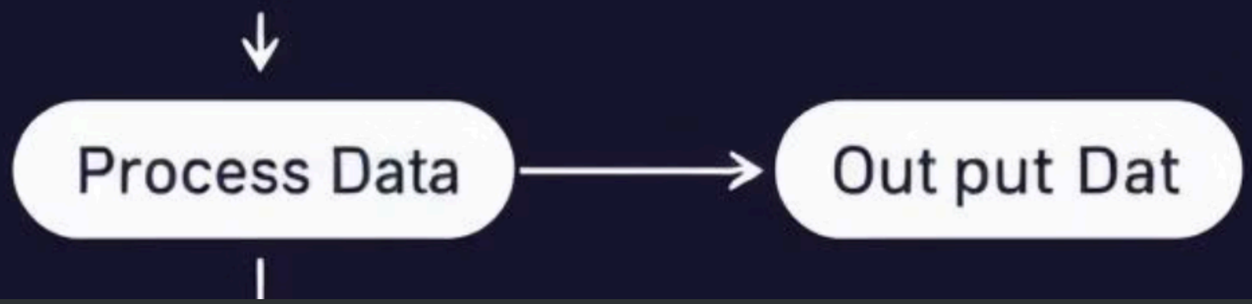


Python Functions and Lambda Expressions

Welcome to a deep dive into Python functions and lambda expressions! This presentation explores their functionalities, syntax, and real-world applications.

by Aravind K



Introduction to Functions

Functions are blocks of reusable code that perform specific tasks.

They promote code organization, avoid repetition, and improve readability.

1 **Definition**

Functions are defined using the 'def' keyword, followed by the function name and parameters.

2 **Purpose**

Functions break down complex tasks into smaller, manageable units.

3 **Syntax**

The basic syntax involves 'def', function name, parameters, and a code block.

Example of a Simple Function

Functions can perform operations like calculations, data processing, or any task you need to repeat.

They can return a result that can be used later in the program.

Function Definition

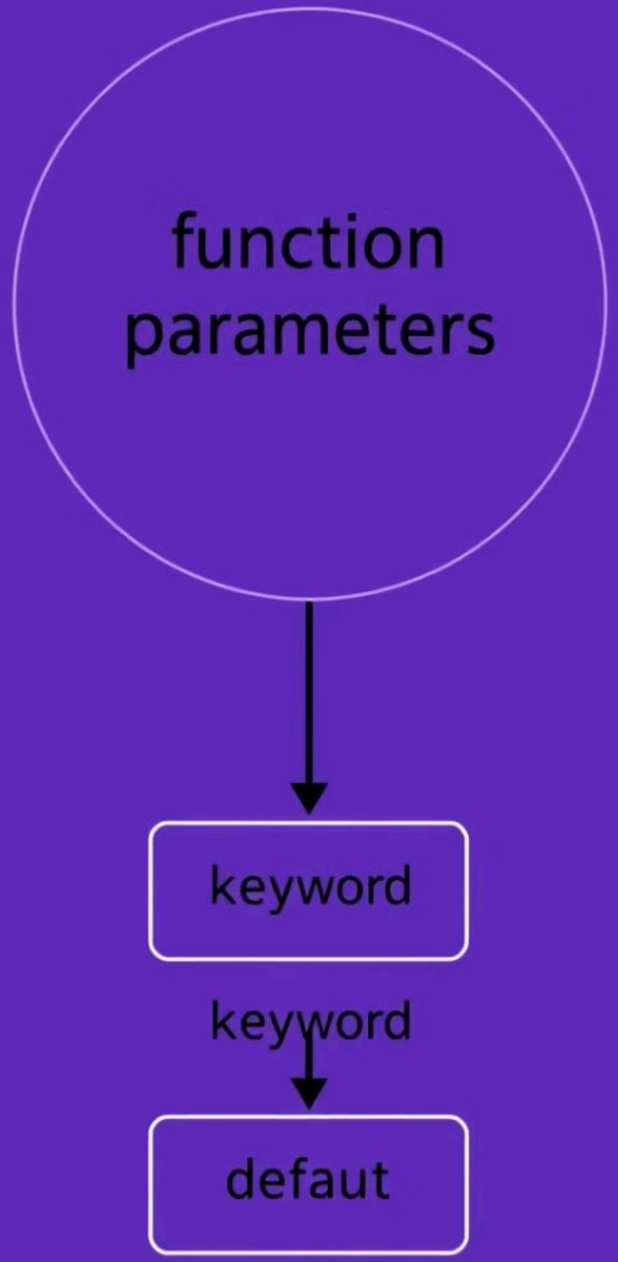
```python
def greet(name):
    return f"Hello, {name}!"
```

Function Call

```python
message = greet("Aravind")
print(message)
```

Output

```
Hello, Aravind!
```



Function Parameters

Parameters allow functions to accept and process different inputs, making them more versatile.

Parameter Type	Description	Example
Positional Parameters	Must be passed in the correct order.	```python def add(x, y): return x + y ```
Keyword Parameters	Passed by explicitly stating the parameter name.	```python def add(x, y): return x + y add(y=5, x=3) ```
Default Parameters	Allow setting default values for optional parameters.	```python def greet(name="User") : return f"Hello, {name}!" ```

Lambda Expressions

Lambda expressions are anonymous functions defined using the 'lambda' keyword.

They are used for short, throwaway functions without the need to formally define them.

1 Definition

A lambda function is defined in a single line using 'lambda', arguments, and an expression.

2 Purpose

Lambda functions provide a concise way to create simple functions without the need for a formal function definition.

3 Use Cases

They are commonly used with functions like 'map', 'filter', and 'reduce' to apply simple operations.

Example of a Lambda Function

Lambda functions are a convenient way to create concise functions.

They are useful for tasks that require simple operations within a larger program.

Lambda Function

```python
square = lambda x: x * x
```

Function Call

```python
result = square(5)
print(result)
```

Output

```
25
```

Using Lambda with Map, Filter, and Reduce

'Map', 'filter', and 'reduce' are higher-order functions that operate on iterables.

Lambda functions are often used with these functions to apply operations efficiently.

1

Map

Apply a lambda function to each item in an iterable and return the results.

2

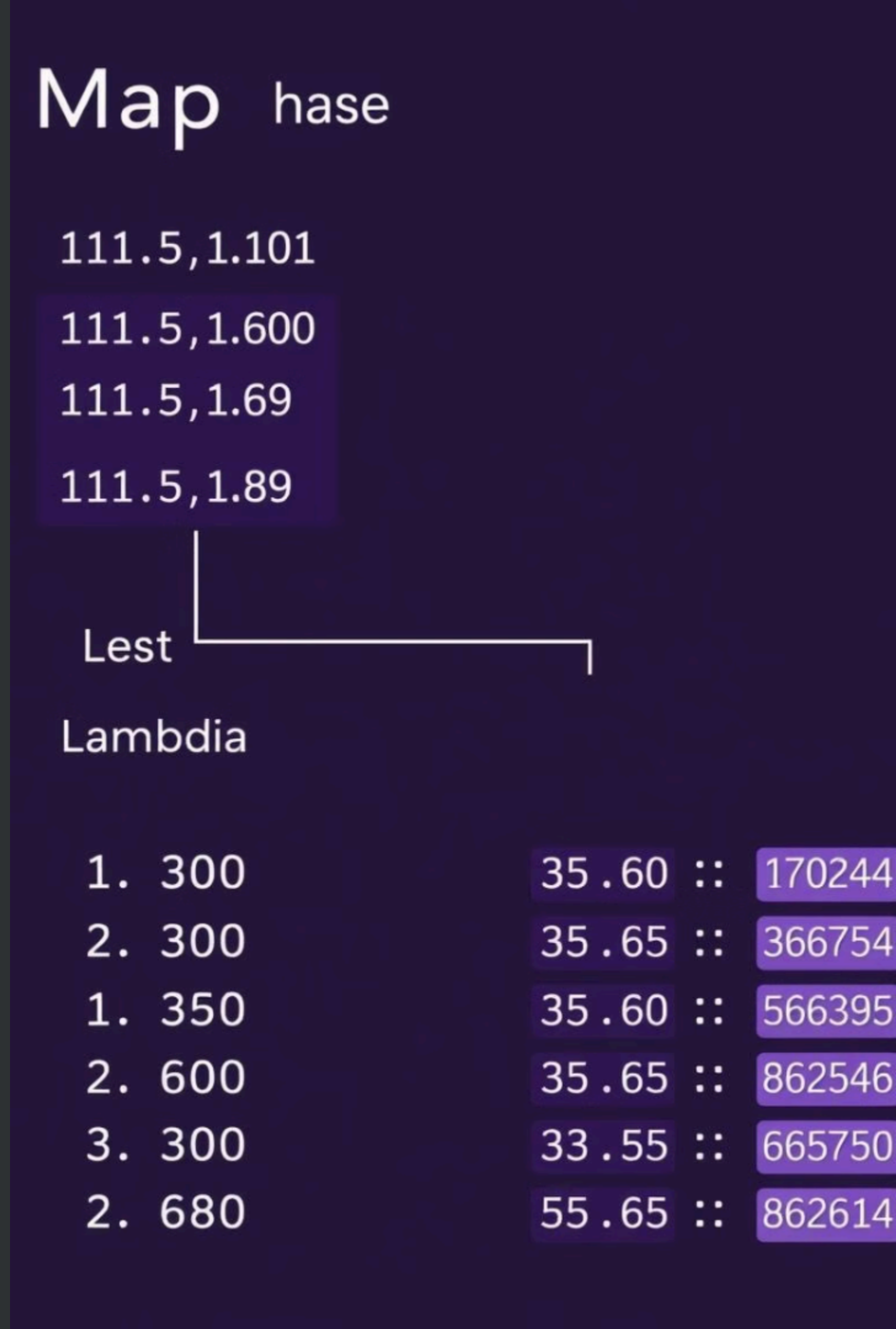
Filter

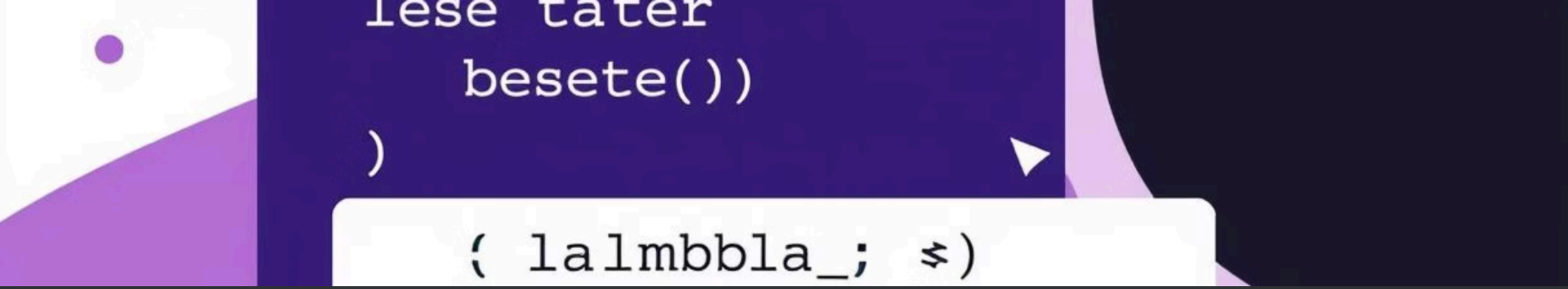
Use a lambda function to filter items in an iterable based on a condition.

3

Reduce

Combine items in an iterable into a single value using a lambda function.





Summary

Functions provide a structured way to organize and reuse code.

Lambda expressions offer a concise syntax for creating anonymous functions, particularly useful for tasks that require simple operations.

f(x)

Functions

Promote code reuse and organization.

λ

Lambda Functions

Provide a concise way to create anonymous functions.

Map, Filter, Reduce

Higher-order functions that work efficiently with lambda expressions.