LAMBDA FUNCTION IN PYTHON

TABLE OF CONTENTS

WHY IT'S IMPORTANT?

Group (B) (ग्रुप -बी)

Q.2 Define cloud computing. Explain benefits of cloud computing in brief. 4
(क्लाउड कंप्यूटिंग को परिभाषित करें। क्लाउड कंप्यूटिंग के लाभों को संक्षेप में वर्णन करें।)

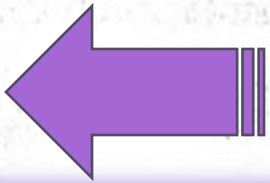
OR (अथवा)

Explain 2-Dimentional array with the help of proper example. 4
(2-आयामी सरणी को उचित उदाहरण की सहायता से वर्णन करें।)

Q.3 Explain LPWAN. Write its characteristics. 4
(यल पी डब्लू ए एन को वर्णन करें। इसकी विशेषताए को लिखें)

OR (अथवा)

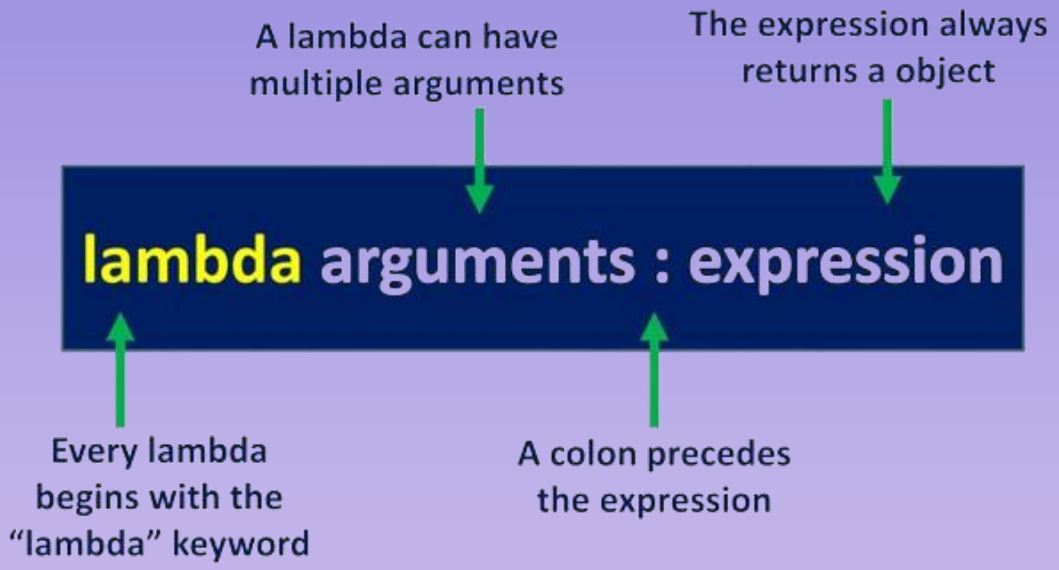
Explain Lambda function and its uses. 4
(लैम्ब्डा फ़ंक्शन और इसके उपयोगों को वर्णन करें।)



WHAT IS LAMBDA FUNCTION?

- A lambda function is a small **anonymous** function that can have any number of parameters, but can only have one expression, which is evaluated and then returned.
- As we already know the **def** keyword is used to define a normal function in Python. Similarly, the **lambda** keyword is used to define an anonymous function in Python.

BASIC SYNTAX



SYNTAX : lambda arguments : expression

EXAMPLES :

- ```
  double = lambda x: x * 2
  result = double(5)
  print(result)      # Output will be 10
  ```
- ```
  add = lambda x, y : x + y
  print(add(3, 5))      # Output: 8
  ```
- ```
  average = lambda x, y, z : (x + y + z) / 3
  print(average(10, 20, 30 ))      # Output: 20
  ```
  ```

HOW IT'S DIFFERENT FROM DEF

```python
def find_even(numbers):
    even_numbers = []
    for num in numbers:
        if num % 2 == 0:
            even_numbers.append(num)
    return even_numbers

numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
result = find_even(numbers)
print("Even numbers using defined function:", result)
```

OUTPUT :

```
Even numbers using defined function: [2, 4, 6, 8, 10]
```

```python
find_even = lambda numbers: [num for num in numbers if num % 2 == 0]

numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
result = find_even(numbers)
print("Even numbers using lambda function:", result)
```

USE CASES

- Lambda functions are generally used in situations where we need a simple function for a short period of time.
- Another common use case is as an argument to a **higher-order function** (a function that takes other functions as an argument.)

Example :

```
cube = lambda x: x * x * x
def addtocube(fx, value):
    return fx(value) + value
print(addtocube(cube, 2))  #Output : 10
```

- Let's explore some of the most common higher-order functions: **map, filter, and reduce**.

1. MAP

- A **map()** function applies a specific function to each item in an iterable (such as a list, tuple, or dictionary) and returns the updated object or iterable .

SYNTAX : **map(**function, iterable**)**

EXAMPLE :

```
numbers = [1, 2, 3, 4, 5]
squared_numbers = map(lambda x: x * x, numbers)

squared_numbers_list = list(squared_numbers)
print(squared_numbers_list)  # Output: [1, 4, 9, 16, 25]
```

2. FILTER

- The **filter()** function return a sequence of elements or object from an iterable that satisfy a specified condition defined by a function, returning only those elements for which the function evaluates to **True**.

SYNTAX : **map(**Boolean function, iterable**)**

EXAMPLE :

```
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

filtered_numbers = filter(lambda x: x > 5, numbers)

filtered_numbers_list = list(filtered_numbers)
print(filtered_numbers_list)  # Output: [6, 7, 8, 9, 10]
```

2. REDUCE

- The **reduce()** function in Python is used to apply a function to the items of an iterable like list, from left to right, to reduce them to a single value.
- It's a part of **funtools** module that must need to be imported.
- The function() must take two arguments and return a single value.

SYNTAX : **reduce(**function, iterable**)**

EXAMPLE :

```
from functools import reduce

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

sum_of_numbers = reduce(lambda x, y: x + y, numbers)
print(sum_of_numbers)       # Output : 15
```

THANKS!

DOES ANYONE HAVE ANY QUESTIONS ?

CREATED BY :

HARSH DEEP