PYTHON

LECTURE 23

Today's Agenda

- **User Defined Functions-IV**
  - Anonymous Functions OR Lambda Function

What Are Anonymous Functions ?

- An **anonymous** function is a function that is ***defined without a name***.

- While **normal functions** are defined using the **def** keyword, we define **anonymous functions** using the **lambda** keyword.

- Hence, **anonymous functions** are also called **lambda** functions.

Syntax Of Lambda Functions

- **Syntax:**

  lambda [arg1,arg2,..]:[expression]

  - ***lambda*** is a keyword/operator and can have any number of arguments.
  - But it can have only one **expression**.
  - Python evaluates the **expression** and returns the result automatically.

What Is An Expression ?

- An **expression** here is anything that can return some value.
- The following items qualify as expressions.
  - **Arithmetic operations** like a+b and a**b
  - **Function calls** like sum(a,b)
  - **A print statement** like print(“Hello”)

So, What Can Be Written In Lambda Expression ?

- **Assignment statements cannot be used in lambda** , because they don't return anything, not even **None** (null).
- Simple things such as **mathematical operations**, **string operations** etc. are OK in a lambda.
- **Function calls** are expressions, so it's OK to put a function call in a lambda, and to pass arguments to that function.
- Even **functions** that return **None**, like the ***print*** function in Python 3, can be used in a lambda.
- **Single line if – else** is also allowed as it also evaluates the condition and returns the result of **true** or **false** expression

How To Create Lambda Functions ?

- Suppose, we want to make a **function** which will ***calculate sum of two numbers.***

- In **normal approach** we will do as shown below:

```
def add(a,b):
        return a+b
```

- In case of **lambda function** we will write it as:

```
lambda a,b: a+b
```

Why To Create Lambda Functions ?

- A very common doubt is that when we can define our functions using **def** keyword , then **why we require lambda functions ?**

- The most common use for **lambda functions** is in code that requires **a simple one-line function**, where it would be an overkill to write a complete **normal function**.

- We will explore it in more detail when we will discuss **two** very important functions in **Python** called **map( )** and **filter( )**

How To Use Lambda Functions ?

- There are 2 ways to use a **Lambda Function**.
  - **Using it anonymously in inline mode**
  - **Using it by assigning it to a variable**

How To Use Lambda Functions ?

- Using it as **anonymous function**

```
print((lambda a,b: a+b)(2,3))
```

Output:

```
5
```

How To Use Lambda Functions ?

- Using it by assigning it to a variable

```
sum=lambda a,b: a+b

print(sum(2,3))
print(sum(5,9))
```

Output:

```
5
14
```

What is happening in this code ?

The statement **lambda a,b:a+b** , is creating a **FUNCTION OBJECT** and returning that object . The variable **sum** is referring to that object. Now when we write **sum(2,3)**, it behaves like function call

Guess The Output ?

```
sum=lambda a,b: a+b

print(type(sum))
print(sum)
```

Since functions also are objects in Python , so they have their a unique memory address as well as their corresponding class as function

Output:

```
<class 'function'>
<function <lambda> at 0x000000000050C1E0>
```

Example

```
squareit=lambda a: a*a

print(squareit(25))
print(squareit(10))
```

Output:

```
625
100
```

Example

```
import math
sqrt=lambda a: math.sqrt(a)

print(sqrt(25))
print(sqrt(10))
```

Output:

```
5.0
3.1622776601683795
```

Exercise

- **Write a lambda function that returns the first character of the string passed to it as argument**

Solution:

```
firstchar=lambda str: str[0]

print("First character of Bhopal :",firstchar("Bhopal"))
print("First character of Sachin :",firstchar("Sachin"))
```

Output:

```
First character of Bhopal : B
First character of Sachin : S
```

Exercise

- **Write a lambda function that returns the last character of the string passed to it as argument**

Solution:

```
lastchar=lambda str: str[len(str)-1]

print("Last character of Bhopal :",lastchar("Bhopal"))
print("Last character of Sachin :",lastchar("Sachin"))
```

Output:

```
Last character of Bhopal : l
Last character of Sachin : n
```

Exercise

- **Write a lambda function that returns True or False depending on whether the number passed to it as argument is even or odd**

Solution:

```
iseven=lambda n: n%2==0
print("10 is even :",iseven(10))
print("7 is even:",iseven(7))
```

Output:

```
10 is even : True
7 is even: False
```

Exercise

- **Write a lambda function that accepts 2 arguments and returns the greater amongst them**

Solution:

```
maxnum=lambda a,b: a if a>b else b
print("max amongst 10 and 20 :",maxnum(10,20))
print("max amongst 15 and 5 :",maxnum(15,5))
```

Output:

```
max amongst 10 and 20 : 20
max amongst 15 and 5 : 15
```