Name: Abhishek Kanoujia

DATA ENGINEERING BATCH 1

DAY 9 ASSIGNMENT

Python Lambda Functions

1. Introduction to Python Lambda Functions:

The tutorial begins by introducing Python lambda functions as anonymous functions lacking a name. The syntax and structure of lambda functions are explained, highlighting their capability to handle a single expression effectively.

```
str1 = 'HexaforHexa'

upper = lambda string: string.upper()

print(upper(str1))
```

2. Condition Checking with Lambda:

An example illustrates the use of lambda functions for condition checking. The format_numeric lambda function formats numeric values based on whether they are integers or floats.

```
format_numeric = lambda num: f"{num:e}" if isinstance(num, int) else f"{num:,.2f}"

print("Int formatting:", format_numeric(1000000))
```

print("Float formatting:", format_numeric(999999.789541235))

3. Comparison with def Functions:

A comparison is made between traditional def functions and lambda functions. The example calculates the cube of a number using both approaches.

```
def cube(y):
    return y*y*y
lambda_cube = lambda y: y*y*y
print("Using def function, cube:", cube(5))
print("Using lambda function, cube:", lambda_cube(5))
```

4. Practical Uses of Lambda Functions:

The tutorial showcases practical applications of lambda functions, including their use in list comprehension, if-else statements, and handling multiple statements.

4.1 Lambda Functions with List Comprehension:

```
is_even_list = [lambda arg=x: arg * 10 for x in range(1, 5)]
for item in is_even_list:
    print(item())
```

4.2 Lambda Functions with if-else:

```
Max = lambda a, b : a if(a > b) else b
```

```
print(Max(1, 2))
```

4.3 Lambda with Multiple Statements:

```
List = [[2,3,4],[1, 4, 16, 64],[3, 6, 9, 12]]
sortList = lambda x: (sorted(i) for i in x)
secondLargest = lambda x, f : [y[len(y)-2] for y in f(x)]
res = secondLargest(List, sortList)
print(res)
```

5. Lambda Functions with Built-in Functions:

The tutorial demonstrates using lambda functions with built-in functions such as filter(), map(), and reduce().

5.1 Using lambda() Function with filter():

```
li = [5, 7, 22, 97, 54, 62, 77, 23, 73, 61]
final_list = list(filter(lambda x: (x % 2 != 0), li))
print(final_list)
```

5.2 Using lambda() Function with map():

```
li = [5, 7, 22, 97, 54, 62, 77, 23, 73, 61]
final_list = list(map(lambda x: x*2, li))
print(final_list)
```

5.3 Using lambda() Function with reduce():

```
from functools import reduce
li = [5, 8, 10, 20, 50, 100]
sum = reduce((lambda x, y: x + y), li)
print(sum)
```

6. Arbitrary Arguments and Keyword Arguments:

The tutorial covers the use of lambda functions with arbitrary positional and keyword arguments, demonstrating their flexibility in handling variable numbers of arguments.

```
def add(*args):
  s = 0
  for x in args:
    s += x
  return s

result = add(10, 20, 30, 40)
print(result)
```

7. Mixed Types of Arguments:

An example is provided where lambda functions handle mixed types of arguments, emphasizing the importance of the order of arguments in the argument list.

```
def percent(math, sci, **optional):
  s = math + sci
  for k, v in optional.items():
    s += v
  return s / (len(optional) + 2)

result = percent(math=80, sci=75, Eng=70, Hist=65, Geo=72)
print("Percentage:", result)
```