

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))
```

```
C:\pythan310\python.exe C:/Users/Abhishek/PycharmProjects/pythonProject1/Lamdafunction.py  
HEXAFORHEXA  
  
Process finished with exit code 0
```

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))
```

```
C:\pythan310\python.exe C:/Users/Abhishek/PycharmProjects/pythonProject1/Lamdafunction.py
Int formatting: 1.000000e+06
Float formatting: 999,999.79

Process finished with exit code 0
```

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))
```

```
C:\pythan310\python.exe C:/Users/Abhishek/PycharmProjects/pythonProject1/Lamdafunction.py
Using def function, cube: 125
Using lambda function, cube: 125

Process finished with exit code 0
```

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())

```
C:\pythan310\python.exe C:/Users/Abhishek/PycharmProjects/pythonProject1/Lamdafunction.py
10
20
30
40

Process finished with exit code 0
```

4.2 Lambda Functions with if-else:

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

print(Max(1, 2))

```
C:\pythan310\python.exe C:/Users/Abhishek/PycharmProjects/pythonProject1/Lamdafunction.py
2

Process finished with exit code 0
```

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)

```
C:\pythan310\python.exe C:/Users/Abhishek/PycharmProjects/pythonProject1/Lamdafunction.py
[3, 16, 9]

Process finished with exit code 0
```

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)
```

```
C:\pythan310\python.exe C:/Users/Abhishek/PycharmProjects/pythonProject1/Lamdafunction.py  
[5, 7, 97, 77, 23, 73, 61]
```

```
Process finished with exit code 0
```

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)
```

```
C:\pythan310\python.exe C:/Users/Abhishek/PycharmProjects/pythonProject1/Lamdafunction.py  
[10, 14, 44, 194, 108, 124, 154, 46, 146, 122]
```

```
Process finished with exit code 0
```

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)
```

```
C:\pythan310\python.exe C:/Users/Abhishek/PycharmProjects/pythonProject1/Lamdafunction.py  
193
```

```
Process finished with exit code 0
```

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)
```

```
C:\pythan310\python.exe C:/Users/Abhishek/PycharmProjects/pythonProject1/Lamdafunction.py
100

Process finished with exit code 0
```

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)

C:\pythan310\python.exe C:/Users/Abhishek/PycharmProjects/pythonProject1/Lamdafunction.py
Percentage: 72.4

Process finished with exit code 0

Reading JSON Strings to Python Dictionaries:

1.Import the json Module:

The json module in Python provides methods for working with JSON data.

```
import json
```

2.JSON String representing a Dictionary:

Suppose you have a JSON string representing a dictionary:

```
json_dict_string = '{"name": "John", "age": 30, "city": "New York"}'
```

3.Use json.loads():

The json.loads() function is used to load (parse) a JSON string and convert it into a Python dictionary.

```
python_dict = json.loads(json_dict_string)
```

4.Accessing Values in the Dictionary:

You can now access values in the resulting Python dictionary as you would with any other dictionary.

```
print("Python Dictionary:", python_dict)
```

```
print("Name:", python_dict["name"])
```

```
print("Age:", python_dict["age"])
```

```
print("City:", python_dict["city"])
```

Reading JSON Strings to Python Lists:

1.JSON String representing a List:

Suppose you have a JSON string representing a list:

```
json_list_string = '[1, 2, 3, 4, 5]'
```

2.Use json.loads():

Similar to the dictionary example, use json.loads() to parse the JSON string and convert it into a Python list.

```
python_list = json.loads(json_list_string)
```

3.Accessing Values in the List:

You can access values in the resulting Python list as you would with any other list.

```
print("\nPython List:", python_list)
```

```
print("Third element:", python_list[2])
```

4.Handling JSONDecodeError:

If the JSON string is malformed or not valid JSON, the json.loads() function will raise a json.JSONDecodeError. It's a good practice to handle this potential error to ensure the robustness of your code.

try:

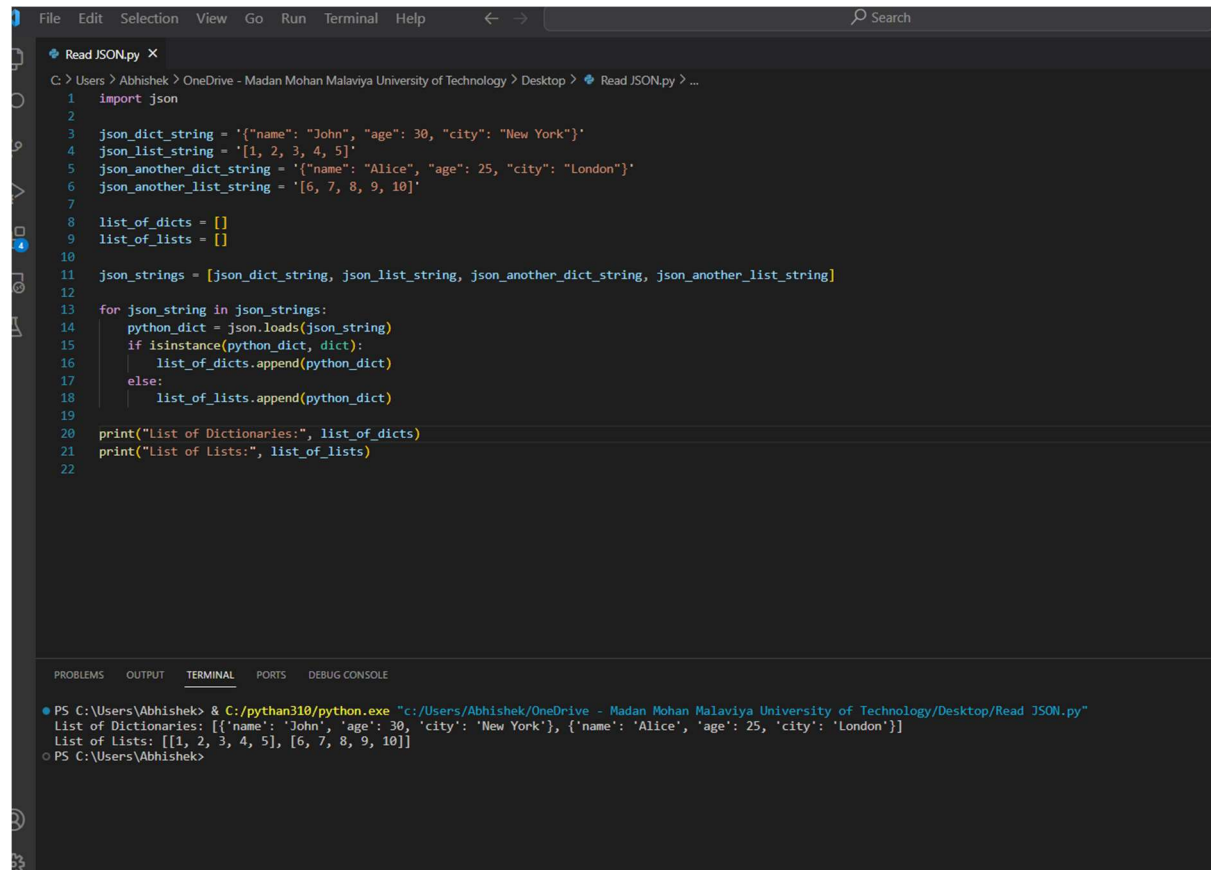
```
python_dict = json.loads(json_dict_string)
```

except json.JSONDecodeError as e:

```
print("Error decoding JSON:", e)
```


This try-except block will catch and print any JSON decoding errors that might occur.

Example:



```
File Edit Selection View Go Run Terminal Help
Read JSON.py X
C:\Users\Abhishek> OneDrive - Madan Mohan Malaviya University of Technology> Desktop> Read JSON.py> ...
1 import json
2
3 json_dict_string = '{"name": "John", "age": 30, "city": "New York"}'
4 json_list_string = '[1, 2, 3, 4, 5]'
5 json_another_dict_string = '{"name": "Alice", "age": 25, "city": "London"}'
6 json_another_list_string = '[6, 7, 8, 9, 10]'
7
8 list_of_dicts = []
9 list_of_lists = []
10
11 json_strings = [json_dict_string, json_list_string, json_another_dict_string, json_another_list_string]
12
13 for json_string in json_strings:
14     python_dict = json.loads(json_string)
15     if isinstance(python_dict, dict):
16         list_of_dicts.append(python_dict)
17     else:
18         list_of_lists.append(python_dict)
19
20 print("List of Dictionaries:", list_of_dicts)
21 print("List of Lists:", list_of_lists)
22
```

PROBLEMS OUTPUT TERMINAL PORTS DEBUG CONSOLE

```
PS C:\Users\Abhishek> & C:/python310/python.exe "c:/Users/Abhishek/OneDrive - Madan Mohan Malaviya University of Technology/Desktop/Read JSON.py"
List of Dictionaries: [{'name': 'John', 'age': 30, 'city': 'New York'}, {'name': 'Alice', 'age': 25, 'city': 'London'}]
List of Lists: [[1, 2, 3, 4, 5], [6, 7, 8, 9, 10]]
PS C:\Users\Abhishek>
```