**Madiha Aimon Tappal**

[**madihaaimon@gmail.com**](mailto:madihaaimon@gmail.com)

**Data Engineering Batch – 1**

**Coding Challenge**

**Python**

* **Execute with one example Lambda Functions in Python**

### Lambda Functions in Python

#### **1. Introduction:**

* In Python, a lambda function, also known as an anonymous function, is a concise way to create small, one-time-use functions without formally defining them using the **def** keyword.
* Lambda functions are often used for short, simple operations where a full function definition seems unnecessary.

#### **2. Syntax:**

* The basic syntax of a lambda function is **lambda arguments: expression**.
* Lambda functions can take any number of arguments but can only have one expression.

# Example of a lambda function to add two numbers

add = lambda x, y: x + y

#### **3. Use Cases:**

* **1. Functional Programming:** Lambda functions are commonly used in functional programming constructs such as **map()**, **filter()**, and **sorted()**.
* **2. Short-lived Operations:** Use lambda functions for operations that are simple and short-lived, where creating a full function definition is overkill.
* **3. Callbacks:** Lambda functions are often used as callbacks in event-driven programming or functions that are passed as arguments to other functions.

#### **4. Example:**

* Suppose you have a list of numbers and you want to create a new list with the squares of each number using **map()** and a lambda function:

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

squared = list(map(lambda x: x\*\*2, numbers))

#### **5. Advantages:**

* **Conciseness:** Lambda functions are concise and can be defined in a single line.
* **Readability:** Useful for short, simple operations, enhancing code readability when used appropriately.
* **No Need for Naming:** Since lambda functions are anonymous, there's no need to come up with a function name.

#### **6. Example Use Cases:**

**Sorting a List of Tuples:**

pairs = [(1, 'one'), (4, 'four'), (3, 'three'), (2,'two')]

sorted\_pairs = sorted(pairs, key=lambda x: x[0])

**Filtering a List:**

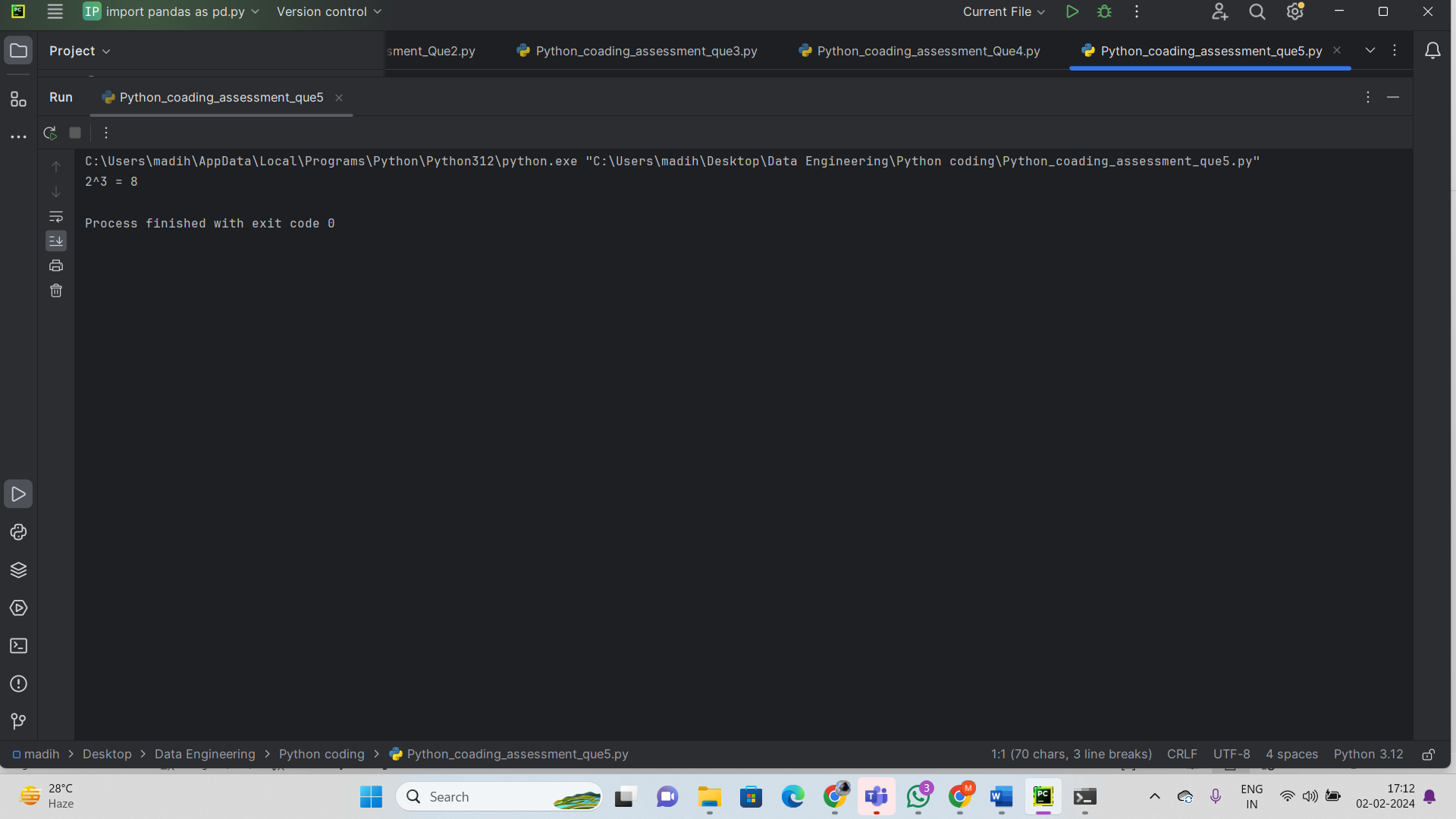
numbers = [1, 2, 3, 4, 5, 6]

even\_numbers = list(filter(lambda x: x % 2 == 0, numbers))

Execution of lambda function Example:

power = lambda x, n: x\*\*n  
result = power(2, 3)  
print("2^3 =", result)

Output:



### Reading JSON Strings to Python dicts or lists

#### **1. JSON (JavaScript Object Notation):**

* JSON is a lightweight data interchange format that is easy for humans to read and write, and easy for machines to parse and generate.
* It consists of key-value pairs and supports nested structures.

#### **2. The** json **Module in Python:**

* The **json** module in Python provides methods for encoding and decoding JSON data.
* The **json.loads()** function is used to decode a JSON-formatted string into a Python object (dict, list, etc.).

#### **3. Decoding JSON using** json.loads()**:**

* The **json.loads()** function takes a JSON-formatted string as an argument and returns a Python object.

import json

# Example JSON string

json\_string = '{"name": "John", "age": 30, "city": "New York"}'

# Decode JSON string to a Python dictionary

data\_dict = json.loads(json\_string)

#### **4. Decoding JSON Arrays:**

* JSON arrays can be decoded into Python lists using **json.loads()**.

import json

# Example JSON array string

json\_array\_string = '[1, 2, 3, 4, 5]'

# Decode JSON array string to a Python list

data\_list = json.loads(json\_array\_string)

The **json.load()** function is used to read JSON data from a file directly into a Python object.

import json

# Example reading JSON data from a file

with open('data.json', 'r') as file:

data\_from\_file = json.load(file)

Example:

import json  
  
# # JSON string representing a dictionary  
# json\_string\_dict = '{"name": "Alice", "age": 30, "city": "New York"}'  
#  
# # Parse JSON string to a Python dictionary  
# python\_dict = json.loads(json\_string\_dict)  
#  
# print("Python Dictionary:", python\_dict)  
# print("Name:", python\_dict['name'])  
# print("Age:", python\_dict['age'])  
# print("City:", python\_dict['city'])  
  
# JSON string representing a list  
json\_string\_list = '[1, 2, 3, 4, 5]'  
  
# Parse JSON string to a Python list  
python\_list = json.loads(json\_string\_list)  
  
print("\nPython List:", python\_list)  
print("Sum of List:", sum(python\_list))

Output:

