1.Optimization:

Optimization in Python refers to improving the efficiency and performance of code, often by reducing runtime, memory usage, or resource consumption. This can be achieved through various techniques, such as using efficient data structures (e.g., lists, dictionaries), avoiding redundant calculations, leveraging built-in functions and libraries like NumPy, or applying algorithmic improvements (e.g., dynamic programming). Profiling tools like Profile or time it help identify bottlenecks. Additionally, Python supports optimization libraries such as SciPy for mathematical optimization tasks, and using compiled extensions (e.g., python) can further enhance performance in computationally heavy code.

2. Different types of optimization:

Ans: 1. Algorithmic Optimization.

2 . Code Optimization.

3. Memory Optimization.

4. Compiler-Level Optimization.

5. Parallelization and Concurrency.

6. Numerical Optimization.

7. Profiling and Benchmarking.

8. Caching and Memorization

Q: Minimize the function in Python :

Source Code:

```
import numpy as np

from scipy.optimize import minimize

# Define the function

def f(x):

    X, Y = x

    return X*2 + Y*2 + 3*X + 4*Y + 5

# Initial guess (X=0, Y=0)

initial_guess = [0, 0]

# Use minimize function to find the minimum

result = minimize(f, initial_guess)
```

```
# Print the result
print("Optimal values for X and Y:", result.x)
print("Minimum value of the function:", result.fun)
```

<u>Output:</u>

```
Optimal values for X and Y: [-1.05022526e+08 -1.26027058e+08]
Minimum value of the function: -1281274974.6586776
```