Best Fit :-

```
def bestFit(blockSize, m, processSize, n):

    # Stores block id of the block
    # allocated to a process
    allocation = [-1] * n

    # pick each process and find suitable
    # blocks according to its size ad
    # assign to it
    for i in range(n):

        # Find the best fit block for
        # current process
        bestIdx = -1
        for j in range(m):
            if blockSize[j] >= processSize[i]:
                if bestIdx == -1:
                    bestIdx = j
                elif blockSize[bestIdx] > blockSize[j]:
                    bestIdx = j

        # If we could find a block for
        # current process
        if bestIdx != -1:

            # allocate block j to p[i] process
            allocation[i] = bestIdx

            # Reduce available memory in this block.
            blockSize[bestIdx] -= processSize[i]
```

```
        print("Process No. Process Size    Block no.")
        for i in range(n):
                print(i + 1, "                  ", processSize[i],
                                                                end = "              ")
                if allocation[i] != -1:
                        print(allocation[i] + 1)
                else:
                        print("Not Allocated")


#       blockSize = [100, 500, 200, 300, 600]
#       processSize = [212, 417, 112, 426]


if __name__ == '__main__':
    blockSize = list(map(int, input("Enter memory block sizes (comma-separated): ").split(',')))
    m = len(blockSize)
    processSize = list(map(int, input("Enter process sizes (comma-separated): ").split(',')))
    n = len(processSize)
    bestFit(blockSize, m, processSize, n)
```

OUTPUT :-

```
Enter memory block sizes (comma-separated): 100, 500, 200, 300, 600
Enter process sizes (comma-separated): 212, 417, 112, 426
Process No. Process Size         Block no.
1                212             4
2                417             2
3                112             3
4                426             5
```

```
Enter memory block sizes (comma-separated): 100, 200, 300, 400, 500
Enter process sizes (comma-separated): 120, 360, 480, 600
Process No. Process Size         Block no.
1                120             2
2                360             4
3                480             5
4                600             Not Allocated
```

First Fit :-

```
def firstFit(blockSize, m, processSize, n):

	# Stores block id of the
	# block allocated to a process
	allocation = [-1] * n

	# Initially no block is assigned to any process

	# pick each process and find suitable blocks
	# according to its size ad assign to it
	for i in range(n):
		for j in range(m):
			if blockSize[j] >= processSize[i]:

				# allocate block j to p[i] process
				allocation[i] = j

				# Reduce available memory in this block.
				blockSize[j] -= processSize[i]

				break

	print(" Process No. Process Size  Block no.")
	for i in range(n):
		print(" ", i + 1, "             ", processSize[i],
					"                    ", end = " ")
		if allocation[i] != -1:
			print(allocation[i] + 1)
		else:
			print("Not Allocated")
```

```
# Driver code
if __name__ == '__main__':
        blockSize = [100, 500, 200, 300, 600]
        processSize = [212, 417, 112, 426]
        m = len(blockSize)
        n = len(processSize)

        firstFit(blockSize, m, processSize, n)
```

OUTPUT : -

```
Process No. Process Size        Block no.
 1               212             3
 2               417             5
 3               112             2
 4               326             4
```

```
Process No. Process Size        Block no.
 1               212             2
 2               417             5
 3               112             2
 4               426             Not Allocated
```

Next Fit :-

```
def NextFit(blockSize, m, processSize, n):

    allocation = [-1] * n
    j = 0
    t = m-1
    # pick each process and find suitable blocks
    # according to its size ad assign to it
    for i in range(n):

        # Do not start from beginning
        while j < m:
            if blockSize[j] >= processSize[i]:

                # allocate block j to p[i] process
                allocation[i] = j

                # Reduce available memory in this block.
                blockSize[j] -= processSize[i]

                # sets a new end point
                t = (j - 1) % m
                break
            if t == j:
                # sets a new end point
                t = (j - 1) % m
                # breaks the loop after going through all memory block
                break

            # mod m will help in traversing the
            # blocks from starting block after
```

```
                    # we reach the end.
                    j = (j + 1) % m

        print("Process No. Process Size Block no.")

        for i in range(n):
            print("\t", i + 1, "\t", processSize[i],end = "\t\t")
            if allocation[i] != -1:
                print(allocation[i] + 1)
            else:
                print("Not Allocated")

# Driver Code
if __name__ == '__main__':
        blockSize = [100,500,200,300,600]
        processSize = [212,417,112,426]
        m = len(blockSize)
        n = len(processSize)

        NextFit(blockSize, m, processSize, n)
```

OUTPUT:-

```
Process No. Process Size Block no.
         1       212            3
         2       317            4
         3       112            4
         4       426            5

Process No. Process Size Block no.
         1       212            2
         2       417            5
         3       112            5
         4       426            Not Allocated
```

Worst Fit :-

```
def worstFit(blockSize, m, processSize, n):

    # Stores block id of the block
    # allocated to a process

    # Initially no block is assigned
    # to any process
    allocation = [-1] * n

    # pick each process and find suitable blocks
    # according to its size ad assign to it
    for i in range(n):

        # Find the best fit block for
        # current process
        wstIdx = -1
        for j in range(m):
            if blockSize[j] >= processSize[i]:
                if wstIdx == -1:
                    wstIdx = j
                elif blockSize[wstIdx] < blockSize[j]:
                    wstIdx = j

        # If we could find a block for
        # current process
        if wstIdx != -1:

            # allocate block j to p[i] process
            allocation[i] = wstIdx
```

```
                    # Reduce available memory in this block.
                    blockSize[wstIdx] -= processSize[i]

        print("Process No. Process Size Block no.")
        for i in range(n):
            print(i + 1, "                ",
                  processSize[i], end = "     ")
            if allocation[i] != -1:
                print(allocation[i] + 1)
            else:
                print("Not Allocated")

if __name__ == '__main__':
        blockSize = [100, 500, 200, 300, 600]
        processSize = [212, 417, 112, 426]
        m = len(blockSize)
        n = len(processSize)

        worstFit(blockSize, m, processSize, n)
```

OUTPUT :-