```
# 1. Sort dictionary by keys
d = {'c': 3, 'a': 1, 'b': 2}
sorted_dict = {k: d[k] for k in sorted(d)}
print(sorted_dict)
```

```
{'a': 1, 'b': 2, 'c': 3}
```

```
# 2. Sum list elements by index range
lst = [10, 20, 30, 40, 50]
start = int(input("Start index: "))
end = int(input("End index: "))
print("Sum:", sum(lst[start:end+1]))
```

```
Start index: 0
End index: 4
Sum: 150
```

```
# 3. Check if 8-bit binary is balanced
def is_balanced(s):
    if s.count('0') == s.count('1'):
        return "balanced"
    return "unbalanced"

print(is_balanced("11001010"))
```

```
balanced
```

```
# 4. Return ratio of two values
def get_ratio(a, b):
    return a / b if b != 0 else "Error"

print(get_ratio(10, 2))
```

```
5.0
```

```
# 5. Calculate final price after discount
def get_final_prices(products):
    return {k: v[0] * (1 - v[1]/100) for k, v in products.items()}

data = {'brand-1': [1000, 10], 'brand-2': [2000, 20]}
print(get_final_prices(data))
```

```
{'brand-1': 900.0, 'brand-2': 1600.0}
```

```
# 6. User class with masked details
class User:
    def __init__(self, uid, mobile, email):
        self.uid = uid
        self.mobile = str(mobile)
        self.email = email

    def display(self):
        masked_mob = "*" * (len(self.mobile)-2) + self.mobile[-2:]
        prefix = self.email.split('@')[0]
        masked_mail = "*" * (len(prefix)-5) + prefix[-5:] + "@gamil.com"
        print(f"User-id: {self.uid}\nMobile no: {masked_mob}\nGmail id: {maske

u = User("admin123", 9876543210, "testuserar923@gamil.com")
u.display()
```

```
User-id: admin123
Mobile no: ********10
Gmail id: ********ar923@gamil.com
```

```
# 7. Return list of capital letters
def get_capitals(s):
    return [c for c in s if 'A' <= c <= 'Z']

print(get_capitals("PyThOn"))
```

```
['P', 'T', 'O']
```

```
# 8. Display k-th column of 3x3 matrix
matrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
k = int(input("Enter column index (0-2): "))
column = [row[k] for row in matrix]
print(column)
```

```
Enter column index (0-2): 1
[2, 5, 8]
```

```
# 9. Convert positive to negative and vice-versa
def swap_signs(lst):
    return [-x for x in lst]

print(swap_signs([10, -20, 30, -40]))
```

```
[-10, 20, -30, 40]
```

```
# 10. Check if matrix multiplication is possible
def can_multiply(shape1, shape2):
    if shape1[1] == shape2[0]:
        return "Possible"
    return "Not Possible"

print(can_multiply((2, 3), (3, 4)))
```

```
Possible
```