```
 1.def check_even_odd(num):
 if num % 2 == 0:
return f"{num} is an even number."
 else:
 return f"{num} is an odd number."
number = int(input("Enter a number: "))
print(check_even_odd(number))
2.def factorial(n):
if n == 0 or n == 1:
 return 1
 else:
 return n * factorial(n - 1)
number = int(input("Enter a number: "))
if number < 0:
print("Factorial is not defined for negative numbers.")
else:
 print(f"Factorial of {number} is {factorial(number)}") 3.def gcd(a, b):
```

```
 while b != 0:

  a, b = b, a % b

 return a

num1 = int(input("Enter the first number: "))

num2 = int(input("Enter the second number: "))

print(f"The GCD of {num1} and {num2} is {gcd(num1, num2)}")

4.def reverse_string(s):

  reversed_str = "

  for char in s:

 reversed_str = char + reversed_str

    return reversed_str

input_str = input("Enter a string: ")

print(f"Reversed string is: {reverse_string(input_str)}")

5.def count_char_frequency(s):

freq_dict = {}

 for char in s:
```

```
 if char in freq_dict:
 freq_dict[char] += 1
 else:
freq_dict[char] = 1
return freq_dict
input_str = input("Enter a string: ")
char_frequency = count_char_frequency(input_str)
print("Character frequencies:")
for char, freq in char_frequency.items():
    print(f"'{char}': {freq}")
6.def is_palindrome(s):
 s = s.replace(" ", "").lower()
 return s == s[::-1]
input_str = input("Enter a string: ")
if is_palindrome(input_str):
print(f"'{input_str}' is a palindrome.")
else:
```

```
print(f"'{input_str}' is not a palindrome.")

 7.def replace_substring(s, old_substring, new_substring):

  return s.replace(old_substring, new_substring)

input_str = input("Enter the original string: ")

old_sub = input("Enter the substring to be replaced: ")

new_sub = input("Enter the new substring: ")

result_str = replace_substring(input_str, old_sub, new_sub)

print(f"Updated string: {result_str}")

8.def replace_substring(s, old_substring, new_substring):

result = "

i = 0

while i < len(s):

if s[i:i+len(old_substring)] == old_substring:

result += new_substring

 i += len(old_substring)

else:
 result += s[i]
  i += 1
return result
input_str = input("Enter the original string: ")
old_sub = input("Enter the substring to be replaced: ")
new_sub = input("Enter the new substring: ")
result_str = replace_substring(input_str, old_sub, new_sub)
print(f"Updated string: {result_str}")
9.def round_to_two_decimals(number):
return round(number, 2)
num = 5.6789
rounded_num = round_to_two_decimals(num)
print(f"The number rounded to 2 decimal places is: {rounded_num}")
10.def float_to_string(number):
    return str(number)
def string_to_float(string):
try:
```

```
return float(string)

except ValueError:

print("The string does not represent a valid float.")

 return None

float_number = 3.14159

string_number = float_to_string(float_number)

print(f"Float to string: {string_number}")

converted_float = string_to_float(string_number)

print(f"String to float: {converted_float}")

11.def add(x, y):

 return x + y

def subtract(x, y):

return x - y

def multiply(x, y):

 return x * y

def divide(x, y):
```

```
if y == 0:
 return "Error! Division by zero.
 return x / y
def main():
print("Select operation:")
print("1. Addition")
print("2. Subtraction")
print("3. Multiplication")
print("4. Division")
choice = input("Enter choice (1/2/3/4): ")
 num1 = float(input("Enter first number: "))
num2 = float(input("Enter second number: "))
if choice == '1':
 print(f"{num1} + {num2} = {add(num1, num2)}")
elif choice == '2':
print(f"{num1} - {num2} = {subtract(num1, num2)}")
elif choice == '3':
```

```
print(f"{num1} * {num2} = {multiply(num1, num2)}")

elif choice == '4':

print(f"{num1} / {num2} = {divide(num1, num2)}")

else:

print("Invalid input")

if

__name__ == "__main__":

    main()

15.def multiply_complex(c1, c2):

return complex(c1) * complex(c2)

c1 = complex(3, 4)

c2 = complex(1, 2)

result = multiply_complex(c1, c2)

print(result)

16.def sum_of_elements(lst):

 return sum(lst)
```

```
numbers = [1, 2, 3, 4, 5]
result = sum_of_elements(numbers)
print(result)
17.def remove_duplicates(lst):
return list(set(lst))
my_list = [1, 2, 2, 3, 4, 4, 5]
print(remove_duplicates(my_list))
18.def bubble_sort(lst):
 n = len(lst)
for i in range(n):
for j in range(0, n-i-1):
if lst[j] > lst[j+1]:
lst[j], lst[j+1] = lst[j+1], lst[j]
return lst
my_list = [64, 25, 12, 22, 11]
print(bubble_sort(my_list))
19.def merge_and_remove_duplicates(list1, list2):
```

```
merged_list = list1 + list2
return list(set(merged_list))
list1 = [1, 2, 3, 4]
list2 = [3, 4, 5, 6]
print(merge_and_remove_duplicates(list1, list2))
 20.def second_largest(lst):
first = second = float('-inf')
for num in lst:
 if num > first:
 second = first
 first = num
elif num > second and num != first:
second = num
return second
my_list = [10, 20, 4, 45, 99]
print(second_largest(my_list))
```

```
21.def find_index(tup, element):
 return tup.index(element)
my_tuple = (10, 20, 30, 40, 50)
print(find_index(my_tuple, 30))
 22.def tuple_to_list(tup):
return list(tup)
def list_to_tuple(lst):
 return tuple(lst)
my_tuple = (1, 2, 3)
my_list = [4, 5, 6]
converted_list = tuple_to_list(my_tuple)
converted_tuple = list_to_tuple(my_list)
print(converted_list)
print(converted_tuple)
23.def element_in_tuple(tup, element):
return element in tup
my_tuple = (1, 2, 3, 4, 5)
```

```
print(element_in_tuple(my_tuple, 3))
24.def count_occurrences(tup, element):
return tup.count(element)
my_tuple = (1, 2, 3, 1, 2, 1)
element = 1
result = count_occurrences(my_tuple, element)
print(result)
25.def concatenate_tuples(*tuples):
result = ()
for tup in tuples:
result += tup
return result
tuple1 = (1, 2)
tuple2 = (3, 4)
tuple3 = (5, 6)
result = concatenate_tuples(tuple1, tuple2, tuple3)
```

```
print(result)

 26.def set_operations(set1, set2):

union = set1 | set2

intersection = set1 & set2

return union, intersection

set_a = {1, 2, 3}

set_b = {2, 3, 4}

union_result, intersection_result = set_operations(set_a, set_b)

print("Union:", union_result)

print("Intersection:", intersection_result)

 27.def is_subset(set1, set2):

 return set1.issubset(set2)

set_a = {1, 2}

set_b = {1, 2, 3, 4}

result = is_subset(set_a, set_b)

print(result)

28.def remove_element(s, element):
```

```
s.discard(element)
my_set = {1, 2, 3, 4}
remove_element(my_set, 3)
print(my_set)
 29.set1 = {1, 2, 3, 4, 5}
set2 = {4, 5, 6, 7, 8}
difference = set1.difference(set2)
print(difference)
30.my_list = [1, 2, 2, 3, 4, 4, 5]
my_set = set(my_list)
print(my_set)
 31.my_dict = {'a': 1, 'b': 2, 'c': 3, 'd': 4}
total_sum = sum(my_dict.values())
print(total_sum)
 32.my_dict = {'b': 2, 'a': 1, 'd': 4, 'c': 3}
sorted_dict = dict(sorted(my_dict.items()))
```

```
print(sorted_dict)
 33.dict1 = {'a': 1, 'b': 2}
dict2 = {'c': 3, 'd': 4}
merged_dict = {**dict1, **dict2}
print(merged_dict)
 34.def merge_dicts(dict1, dict2):
merged = dict1.copy()
merged.update(dict2)
return merged
dict1 = {'a': 1, 'b': 2}
dict2 = {'b': 3, 'c': 4}
result = merge_dicts(dict1, dict2)
print(result)
 35.def invert_dict(original):
return {value: key for key, value in original.items()}
original_dict = {'a': 1, 'b': 2, 'c': 3}
inverted_dict = invert_dict(original_dict)
```

```
print(inverted_dict)
```