In [2]:
```
# Q1: What is the output of following expression
#     5 + 4 * 9 % (3 + 1) / 6 - 1
```

In [3]:
```
5 + 4 * 9 % (3 + 1) / 6 - 1
```

Out[3]:
```
4.0
```

In [4]:
```
# Q2: Write a program to check if a Number is Odd or Even. Take number as a input f
```

In [9]:
```
a=5
if a%2==0:
    print('a is even')
else:
    print('a is odd')
```

```
a is odd
```

In [10]:
```
# Q3: Write a program to display the multiplication table by taking a number as inp
#     [Hint : Use print statement inside of a loop]
```

In [11]:
```
for var in range(11):
    print("5*",var,"=",5*var)
```

```
5* 0 = 0
5* 1 = 5
5* 2 = 10
5* 3 = 15
5* 4 = 20
5* 5 = 25
5* 6 = 30
5* 7 = 35
5* 8 = 40
5* 9 = 45
5* 10 = 50
```

In [ ]:
```
# Q4: Write a program which will find all numbers between 2000 and 3200 which are d
#     but are not a multiple of 5.

# Note: The numbers obtained should be printed in a comma-separated sequence on a s
```

In [13]:
```
numbers = []

for i in range(2000, 3201):

    if i % 7 == 0 and i % 5 != 0:
        # If the condition is met, append the number to the list
        numbers.append(i)

print(','.join(map(str, numbers)))
```

```
2002,2009,2016,2023,2037,2044,2051,2058,2072,2079,2086,2093,2107,2114,2121,2128,21
42,2149,2156,2163,2177,2184,2191,2198,2212,2219,2226,2233,2247,2254,2261,2268,228
2,2289,2296,2303,2317,2324,2331,2338,2352,2359,2366,2373,2387,2394,2401,2408,2422,
2429,2436,2443,2457,2464,2471,2478,2492,2499,2506,2513,2527,2534,2541,2548,2562,25
69,2576,2583,2597,2604,2611,2618,2632,2639,2646,2653,2667,2674,2681,2688,2702,270
9,2716,2723,2737,2744,2751,2758,2772,2779,2786,2793,2807,2814,2821,2828,2842,2849,
2856,2863,2877,2884,2891,2898,2912,2919,2926,2933,2947,2954,2961,2968,2982,2989,29
96,3003,3017,3024,3031,3038,3052,3059,3066,3073,3087,3094,3101,3108,3122,3129,313
6,3143,3157,3164,3171,3178,3192,3199
```

In [ ]:
```
Q5: Count the elements of each datatype inside the list and display in output
    [2, 3, 'Py', '10', 1, 'SQL', 5.5, True, 3, 'John', None, 7]
```

In [17]:
```
m_list = [2, 3, 'Py', '10', 1, 'SQL', 5.5, True, 3, 'John', None, 7]

int_count = 0
float_count = 0
str_count = 0
bool_count = 0
none_count = 0
other_count = 0

for item in m_list:
    if isinstance(item, int):
        int_count += 1
    elif isinstance(item, float):
        float_count += 1
    elif isinstance(item, str):
        str_count += 1
    elif isinstance(item, bool):
        bool_count += 1
    elif item is None:
        none_count += 1
    else:
        other_count += 1

print("Integer Count:", int_count)
print("Float Count:", float_count)
print("String Count:", str_count)
print("Boolean Count:", bool_count)
print("None Count:", none_count)
print("Other Datatypes Count:", other_count)
```

```
Integer Count: 6
Float Count: 1
String Count: 4
Boolean Count: 0
None Count: 1
Other Datatypes Count: 0
```

In [ ]:
```
Q6: Add all values from the list with numeric datatypes
    [2, 3, 'Py', '10', 1, 'SQL', 5.5, True, 3, 'John', None, 7]
```

In [21]:
```
my_list = [2, 3, 'Py', '10', 1, 'SQL', 5.5, True, 3, 'John', None, 7]

total_sum = 0
for item in my_list:
    if isinstance(item,int) or isinstance(item ,float):
        total_sum +=item

print("Sum of numeric values in the list:", total_sum)
```

```
Sum of numeric values in the list: 22.5
```

In [ ]:
```
# Q7: Concat all str datatypes with hyphen as a delimiter
#     [2, 3, 'Py', '10', 1, 'SQL', 5.5, True, 3, 'John', None, 7]
```

In [22]:
```
my_list = [2, 3, 'Py', '10', 1, 'SQL', 5.5, True, 3, 'John', None, 7]

concatenated_str = ''

for item in my_list:
    if isinstance(item, str):
        if concatenated_str:
            concatenated_str += '-' + item
        else:
            concatenated_str = item

print("Concatenated string with hyphen as delimiter:", concatenated_str)
```

```
Concatenated string with hyphen as delimiter: Py-10-SQL-John
```

In [ ]:

```
# Q8: Write a UDF that takes list as input and returns sum of all numbers
#     (exclude bool) and count of all str
#     [2, 3, 'Py', '10', 1, 'SQL', 5.5, True, 3, 'John', None, 7]

# Hint:
# -----
# def my_func:
#     # your code

# my_func(l1)
# # output --> {'Sum': xxx, 'Count_of_Strs': xxx}
```

In [26]:

```
l1=  [2, 3, 'Py', '10', 1, 'SQL', 5.5, True, 3, 'John', None, 7]
def my_func(input_list):
    total_sum = 0
    str_count = 0

    for item in input_list:
        if isinstance(item, (int, float)):
            total_sum += item
        elif isinstance(item, str):
            str_count += 1

    return {'Sum': total_sum, 'Count_of_Strs': str_count}
my_func(l1)
```

Out[26]:

```
{'Sum': 22.5, 'Count_of_Strs': 4}
```

In [ ]:

```
Q9: Get only odd numbers from the following list and store the numbers in new list
    li = [5, 7, 22, 97, 54, 62, 77, 23, 73, 61]

    i. Use loops to get the answer
   ii. Use list comprehensions
  iii. Use lambda function with filter
```

In [27]:

```
li = [5, 7, 22, 97, 54, 62, 77, 23, 73, 61]

odd_numbers = []

for num in li:
    if num % 2 != 0:
        odd_numbers.append(num)

print("Using loops:", odd_numbers)
```

```
Using loops: [5, 7, 97, 77, 23, 73, 61]
```

In [28]:

```
li = [5, 7, 22, 97, 54, 62, 77, 23, 73, 61]

odd_numbers = [num for num in li if num % 2 != 0]

print("Using list comprehensions:", odd_numbers)
```

```
Using list comprehensions: [5, 7, 97, 77, 23, 73, 61]
```

In [29]:
```
li = [5, 7, 22, 97, 54, 62, 77, 23, 73, 61]

odd_numbers = list(filter(lambda x: x % 2 != 0, li))

print("Using lambda function with filter:", odd_numbers)
```
```
Using lambda function with filter: [5, 7, 97, 77, 23, 73, 61]
```

In [ ]:
```
Q10: Write a UDF to return the descriptives [sum, count, min, mean, max] for a list
    numbers.
```

In [31]:
```
def calculate_descriptives(numbers):
    if not numbers:
        return None

    total_sum = sum(numbers)
    count = len(numbers)
    minimum = min(numbers)
    mean = total_sum / count
    maximum = max(numbers)

    return {
        "sum": total_sum,
        "count": count,
        "min": minimum,
        "mean": mean,
        "max": maximum
    }

input_numbers = [5, 10, 15, 20, 25]
result = calculate_descriptives(input_numbers)
print(result)
```
```
{'sum': 75, 'count': 5, 'min': 5, 'mean': 15.0, 'max': 25}
```

In [ ]:
```
# Q11: Write an udf to calculate the area of different shapes

# Take shape and dimensions as arguments to udf as follows :

# 1. square which has side
# 2. rectangle which has length and width
# 3. circle which has radius

# The shape should be a positional argument and it's dimensions are taken as kwargs

# Perform proper validation for the user inputs and then calculate area.

# E.g. if shape is square, ensure kwargs has "side" and if so, then you may return
```

In [32]:
```
import math

def calculate_area(shape, **kwargs):
    if shape == "square":
        if "side" in kwargs:
            side = kwargs["side"]
            if side <= 0:
                return "Side length must be a positive number"
            return side ** 2
        else:
            return "Please enter 'side' for a square"

    elif shape == "rectangle":
        if "length" in kwargs and "width" in kwargs:
            length = kwargs["length"]
            width = kwargs["width"]
            if length <= 0 or width <= 0:
                return "Length and width must be positive numbers"
            return length * width
        else:
            return "Please enter 'length' and 'width' for a rectangle"

    elif shape == "circle":
        if "radius" in kwargs:
            radius = kwargs["radius"]
            if radius <= 0:
                return "Radius must be a positive number"
            return math.pi * radius ** 2
        else:
            return "Please enter 'radius' for a circle"

    else:
        return "Unsupported shape"

# Example usage:
print(calculate_area("square", side=5))
print(calculate_area("rectangle", length=5, width=3))
print(calculate_area("circle", radius=7))
```

```
25
15
153.93804002589985
```

In [ ]:

```
# Q12: Write a UDF to reconcile the values within two lists.
#     l1 = ['January', 'February', 'March', 'May', 'June', 'September', 'December']
#     l2 = ['January', 'February', 'April', 'June', 'October', 'December']

# Hint:
# -----
# def func(l1, l2):
#     your code here...

# Output:
# {'Matched': ['January', 'February', 'June', 'December'],
#     'Only in l1': ['March', 'May', 'September'],
#         'Only in l2': ['April', 'October']}
```

In [33]:

```
def reconcile_lists(l1, l2):
    matched = []
    only_in_l1 = []
    only_in_l2 = []

    for item in l1:
        if item in l2:
            matched.append(item)
        else:
            only_in_l1.append(item)

    for item in l2:
        if item not in l1:
            only_in_l2.append(item)

    return {
        'Matched': matched,
        'Only in l1': only_in_l1,
        'Only in l2': only_in_l2
    }
```

```
l1 = ['January', 'February', 'March', 'May', 'June', 'September', 'December']
l2 = ['January', 'February', 'April', 'June', 'October', 'December']

print(reconcile_lists(l1, l2))
```

```
{'Matched': ['January', 'February', 'June', 'December'], 'Only in l1': ['March',
'May', 'September'], 'Only in l2': ['April', 'October']}
```

In [ ]:

```
# Q13: write a UDF to check if a number is prime or not.
```

In [34]:

```
def is_prime(number):
    if number <= 1:
        return False
    elif number <= 3:
        return True
    elif number % 2 == 0 or number % 3 == 0:
        return False
    i = 5
    while i * i <= number:
        if number % i == 0 or number % (i + 2) == 0:
            return False
        i += 6
    return True

print(is_prime(7))
print(is_prime(14))
```

```
True
False
```

In [ ]:

```
# Q14. Write a program which can compute the factorial of a given numbers.
#   The results should be printed in a comma-separated sequence on a single line.
# input() function can be used for getting user(console) input


#Suppose the input is supplied to the program:   8
#Then, the output should be:  40320
#Hints: In case of input data being supplied to the question, it should be assumed
```

In [35]:

```
def factorial(n):
    if n == 0:
        return 1
    else:
        return n * factorial(n-1)

def main():
    num = int(input("Enter a number to compute its factorial: "))
    result = factorial(num)
    print(result)

if __name__ == "__main__":
    main()
```

```
Enter a number to compute its factorial: 67
3647111091818868528824985909660546442716763531404952459370162850026796243694387200
0000000000000
```

In [ ]:

```
# Q15. With a given integral number n, write a program to generate a dictionary tha

#Suppose the following input is supplied to the program: 8
#Then, the output should be: {1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64}
#Hints: In case of input data being supplied to the question, it should be assumed
```

In [36]:
```
def generate_square_dict(n):
    square_dict = {}
    for i in range(1, n+1):
        square_dict[i] = i * i
    return square_dict

def main():
    n = int(input("Enter an integral number (n): "))
    result = generate_square_dict(n)
    print(result)

if __name__ == "__main__":
    main()
```

```
Enter an integral number (n): 5
{1: 1, 2: 4, 3: 9, 4: 16, 5: 25}
```

In [ ]:
```
# Q16. Write a program which accepts a sequence of comma-separated numbers from cor
#Suppose the following input is supplied to the program: 34,67,55,33,12,98
    #Then, the output should be: ['34', '67', '55', '33', '12', '98'] ('34', '67',

#Hints: In case of input data being supplied to the question, it should be assumed
```

In [37]:
```
def main():
    input_str = input("Enter a sequence of comma-separated numbers: ")
    numbers_list = input_str.split(',')
    numbers_tuple = tuple(numbers_list)
    print(numbers_list, numbers_tuple)

if __name__ == "__main__":
    main()
```

```
Enter a sequence of comma-separated numbers: 1,2,3,4,5,6,
['1', '2', '3', '4', '5', '6', ''] ('1', '2', '3', '4', '5', '6', '')
```

In [ ]:
```
# Q17. Write a program that accepts a comma separated sequence of words as input ar
# prints the words in a comma-separated sequence after sorting them alphabetically.

# Suppose the following input is supplied to the program: without,hello,bag,world
# Then, the output should be: bag,hello,without,world

#Hints: In case of input data being supplied to the question, it should be assumed
```

In [38]:
```
def main():
    input_str = input("Enter a comma-separated sequence of words: ")
    words_list = input_str.split(',')
    sorted_words = sorted(words_list)
    sorted_str = ','.join(sorted_words)
    print(sorted_str)

if __name__ == "__main__":
    main()
```

```
Enter a comma-separated sequence of words: red,white,green,yellow
green,red,white,yellow
```

In [ ]:
```
# Q18. Write a program that accepts a sequence of whitespace separated words
# as input and prints the words after removing all duplicate words and sorting then
# Suppose the following input is supplied to the program: hello world and practice
# Then, the output should be: again and hello makes perfect practice world

#Hints: In case of input data being supplied to the question, it should be assumed
#We use set container to remove duplicated data automatically and then use sorted()
```

In [39]:
```
def main():
    input_str = input("Enter a sequence of whitespace-separated words: ")
    words_list = input_str.split()
    unique_words = sorted(set(words_list))
    result = ' '.join(unique_words)
    print(result)

if __name__ == "__main__":
    main()
```

```
Enter a sequence of whitespace-separated words: red green yellow
green red yellow
```

In [ ]:
```
# Q19. Write a program that accepts a sentence and calculate the number of upper ca
# letters and lower case letters.
#Suppose the following input is supplied to the program: Hello world!
#Then, the output should be: UPPER CASE 1 LOWER CASE 9

#Hints: In case of input data being supplied to the question, it should be assumed
```

In [41]:
```
def count_case(sentence):
    upper_count = 0
    lower_count = 0
    for char in sentence:
        if char.isupper():
            upper_count += 1
        elif char.islower():
            lower_count += 1
    return upper_count, lower_count

def main():
    sentence = input("Enter a sentence: ")
    upper_count, lower_count = count_case(sentence)
    print("UPPER CASE", upper_count, "LOWER CASE", lower_count)

if __name__ == "__main__":
    main()
```

```
Enter a sentence: Hello World Where are You
UPPER CASE 4 LOWER CASE 17
```

In [ ]:
```
# Q20. Write a program that takes a string and returns reversed string. i.e. if inp
```

In [44]:
```
def reverse_string(input_str):
    return input_str[::-1]

def main():
    input_str = input("Enter a string: ")
    reversed_str = reverse_string(input_str)
    print("Reversed string:", reversed_str)

if __name__ == "__main__":
    main()
```

```
Enter a string: "where are you going"
Reversed string: "gniog uoy era erehw"
```

In [ ]:

In [ ]: