Python Assignment Questions

Functions:

In [11]:
```python
# Write a function called add_numbers that takes two integers as input and returns their sum.
def add_2_num():
    a = int(input("Enter 1st number:"))
    b = int(input("Enter 2nd number:"))
    Total = a + b
    return print("Total:", Total)

add_2_num()
```
```
Total: 10
```

In [12]:
```python
# Write a function called multiply_numbers that takes two integers as input and returns their product.
def mul_2_num():
    a = int(input("Enter 1st number:"))
    b = int(input("Enter 2nd number:"))
    Total = a * b
    return print("Total:", Total)

mul_2_num()
```
```
Total: 25
```

In [24]:
```python
# Write a function called calculate_average that takes a list of numbers as input and returns the average of those numbers.
def avg_list(nums):
    if len(nums) == 0:
        return 0
    return sum(nums) / len(nums)

num = eval(input("Enter numbers as a list, include commas:"))
avg = avg_list(num)
print("Average of list:",avg)
```
```
Average of list: 7.0
```

In [27]:
```python
# Write a function called is_even that takes an integer as input and returns True if the number is even, and False otherwise.
def is_even():
    a = int(input("Enter a Number:"))
    if a%2 == 0:
        print("True")
    else:
        print("False")

is_even()
```
```
False
```

In [30]:
```python
# Write a function called reverse_string that takes a string as input and returns the reverse of that string.
def rev_string():
    st = input("Enter a string:")
    for words in st:
        words = st[::-1]
        print(words)
        break

rev_string()
```
```
HsAy
```

In [34]:
```python
# Write a function called count_vowels that takes a string as input and returns the number of vowels (a, e, i, o, u) in that string.
def vow_count():
    y = input("Enter a string:")
    v = ('aeiouAEIOU')
    c = 0
    for ch in y:
        if ch in v:
            c +=1
    print("No.of Vowels:", c)

vow_count()
```
```
No.of Vowels: 10
```

In [37]:
```python
# Write a function called find_max that takes a list of numbers as input and returns the maximum value in that list.
def max_num(nums):
    if len(nums) == 0:
        return 0
    return max(nums)

num = eval(input("Enter numbers as a list, include commas:"))
max_nums = max_num(num)
print("Maximum Value in list:",max_nums)
```
```
Maximum Value in list: 88
```

In [41]:
```python
# Write a function called find_min that takes a list of numbers as input and returns the minimum value in that list.
def min_num(nums):
    if len(nums) == 0:
        return 0
    return min(nums)

num = eval(input("Enter numbers as a list, include commas:"))
min_nums = min_num(num)
print("Minimum Value in list:",min_nums)
```
```
Minimum Value in list: 33
```

In [54]:
```python
# Write a Python function to find the maximum of three numbers.
def maxl_num(a, b, c):
        if a >= b and a >= c:
            return a
        elif b >= a and b >= c:
            return b
        else:
            return c


x = int(input("Enter 1st number: "))
y = int(input("Enter 2nd number: "))
z = int(input("Enter 3rd number: "))

max_of_three = maxl_num(x, y, z)
print("Maximum of the three numbers:", max_of_three)
```
```
Maximum of the three numbers: 55
```

In [55]:
```python
# Write a Python function to sum all the numbers in a list.
def sum_of_list(nums2):
    if len(nums2) == 0:
        return 0
    return sum(nums2)

lst1 = eval(input("Enter list of numbers include commas:"))
x = sum_of_list(lst1)
print("Sum of given list of numbers:",x)
```
```
Sum of given list of numbers: 15
```

In [59]:
```python
# Write a Python function to check whether a number falls within a given range.
def check_range():
    x = int(input("Enter Starting range:"))
    y = int(input("Enter Ending range:"))
    inp = int(input("Enter the number to check:"))
    if inp in range(x, y):
        print(f"Yes, {inp} falls within given range")
    else:
        print(f"No, {inp} not falls within given range")

check_range()
```
```
No, 55 not falls within given range
```

In [61]:
```python
# Write a Python function that takes a list and returns a new list with distinct elements from the first list.
def uniq_elem(inp_list):
    uniq_list = []
    for x in inp_list:
        if x not in uniq_list:
            uniq_list.append(x)
    return uniq_list

lst2 = [1,2,3,4,5,5,6,7,3,3,23,4,3,2]
re = uniq_elem(lst2)
print(re)
```
```
[1, 2, 3, 4, 5, 6, 7, 23]
```

In [66]:
```python
# Write a Python function that accepts a string and counts the number of upper and lower case letters.
def count_case(inp):
    count_upper = 0
    count_lower = 0
    for char in inp:
        if char.isupper():
            count_upper += 1
        elif char.islower():
            count_lower += 1
    return count_upper, count_lower

st2 = "AlL ARe Same"
upper, lower = count_case(st2)
print("Upper:", upper)
print("Lower:", lower)
```
```
Upper: 5
Lower: 5
```

Built-in Functions (Lambda, Map, Filter, Reduce):

In [67]:
```python
# Create a lambda function that takes two arguments and returns their sum.
add = lambda a, b: a + b
result = add(10, 5)
print("Sum:", result)
```
```
Sum: 15
```

In [68]:
```python
# Create another lambda function that takes a single argument and returns its square.
square = lambda a: a**2
result = square(5)
print("Square of 5:",result)
```
```
Square of 5: 25
```

In [69]:
```python
# Use the lambda functions to calculate the sum and square of various numbers and print the results.
square = lambda a: a**2
result = square(5)
print("Square of 5:",result)
print("Square of 4:", square(4))
print("Square of 9:", square(9))
print("Square of -2:", square(-2))

add = lambda a, b: a + b
result = add(10, 5)
print("Sum:", result)
print("Sum of 3 and 5:", add(3, 5))
print("Sum of 10 and 7:", add(10, 7))
```
```
Square of 5: 25
Square of 4: 16
Square of 9: 81
Square of -2: 4
Sum: 15
Sum of 3 and 5: 8
Sum of 10 and 7: 17
```

In [70]:
```python
# Say, a = 27, b = 15. Create a lambda function that multiplies argument a with argument b and prints the result.
mul = lambda a,b : a*b
a = 27
b = 15
print("Multiplication of 27, 15:", mul(a,b))
```
```
Multiplication of 27, 15: 405
```

In [71]:
```python
# Define a list of integers. Use the filter() function to create a new list that contains only the even numbers from the original list.
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

even_numbers = list(filter(lambda x: x % 2 == 0, numbers))

print("Even numbers:", even_numbers)
```
```
Even numbers: [2, 4, 6, 8, 10]
```

In [72]:
```python
# Use the filter() function again to create a new list that contains only numbers greater than 10 from the original list. Print both filtered lists.

numbers = [4, 11, 8, 15, 3, 22, 10, 7, 13, 6]

even_numbers = list(filter(lambda x: x % 2 == 0, numbers))

greater_than_10 = list(filter(lambda x: x > 10, numbers))

print("Even numbers:", even_numbers)
print("Numbers greater than 10:", greater_than_10)
```
```
Even numbers: [4, 8, 22, 10, 6]
Numbers greater than 10: [11, 15, 22, 13]
```

In [74]:
```python
# Using filter() with a lambda function, filter even numbers from a given list: num = [17, 20, 32, 45, 53, 61, 72, 84, 97]
num = [17, 20, 32, 45, 53, 61, 72, 84, 97]
even_numbers = list(filter(lambda x: x % 2 == 0, num))
print(even_numbers)
```
```
[20, 32, 72, 84]
```

In [79]:
```python
# Define a list of strings where each string represents a number (e.g., ["1", "2", "3"]).
# Use the map() function to convert each string to an integer and store the result in a new list.

string_numbers = ["1", "2", "3", "10", "25"]

int_numbers = list(map(int, string_numbers))

print("Converted list:", int_numbers)
print("Converted list:", type(int_numbers[2]))
```
```
Converted list: [1, 2, 3, 10, 25]
Converted list: <class 'int'>
```

In [80]:
```python
# Use the map() function again to square each integer in the new list and store the result in another new list. Print both new lists.

string_numbers = ["1", "2", "3", "10", "25"]

int_numbers = list(map(int, string_numbers))

squared_numbers = list(map(lambda x: x ** 2, int_numbers))

print("Integer list:", int_numbers)
print("Squared list:", squared_numbers)
```
```
Integer list: [1, 2, 3, 10, 25]
Squared list: [1, 4, 9, 100, 625]
```

In [83]:
```python
# Create a list of dictionaries, where each dictionary represents a person with keys "name" and "age".
# Example: [{"name": "Alice", "age": 30}, {"name": "Bob", "age": 25}, ...]

pe = [
    {"name": "Yash", "age": 30},
    {"name": "Raju", "age": 25},
    {"name": "Sandy", "age": 35},
    {"name": "Atish", "age": 28},
    {"name": "Ramesh", "age": 22}
]

for person in pe:
    print(f"Name: {person['name']}, Age: {person['age']}")
```
```
Name: Yash, Age: 30
Name: Raju, Age: 25
Name: Sandy, Age: 35
Name: Atish, Age: 28
Name: Ramesh, Age: 22
```

In [89]:
```python
# Use the filter() function to create a new list that contains only the people who are older than 25.
p = [
    {"name": "Yash", "age": 30},
    {"name": "Raju", "age": 25},
    {"name": "Sandy", "age": 35},
    {"name": "Atish", "age": 28},
    {"name": "Ramesh", "age": 22}
]

for person in p:
    if person["age"] > 25:
        print(f"Name: {person['name']}, Age: {person['age']}")
```
```
Name: Yash, Age: 30
Name: Sandy, Age: 35
Name: Atish, Age: 28
```

In [90]:
```python
# Use the map() function to create a new list that contains the names of the filtered people. Print the list of names.
p = [
    {"name": "Yash", "age": 30},
    {"name": "Raju", "age": 25},
    {"name": "Sandy", "age": 35},
    {"name": "Atish", "age": 28},
    {"name": "Ramesh", "age": 22}
]

filtered_people = list(filter(lambda person: person["age"] > 25, p))

names = list(map(lambda person: person["name"], filtered_people))

print("Names of people older than 25:", names)
```
```
Names of people older than 25: ['Yash', 'Sandy', 'Atish']
```

Create a program that reads a list of numbers from the user (you can use the input() function for this). Use map() and filter() functions to:

- Square each number.
- Filter out numbers that are not divisible by 3.
- Print the result.

In [92]:
```python
user_input = input("Enter numbers separated by space: ")

numbers = list(map(int, user_input.split()))

squared_numbers = list(map(lambda x: x ** 2, numbers))

div_by_3 = list(filter(lambda x: x % 3 == 0, squared_numbers))

print("Squared numbers divisible by 3:", div_by_3)
```
```
Squared numbers divisible by 3: [36, 81, 81]
```

In [94]:
```python
# Use map() with a lambda function to square each element of the given list: list1 = [32,45,67,83,2,4,8,10]
list1 = [32,45,67,83,2,4,8,10]
squared_numbers = list(map(lambda x: x ** 2, list1))
print(squared_numbers)
```
```
[1024, 2025, 4489, 6889, 4, 16, 64, 100]
```

In [95]:
```python
# Find the product of all numbers in the list using reduce() and lambda functions: nums = [15, 2, 31, 4, 56]
from functools import reduce

nums = [15, 2, 31, 4, 56]

product = reduce(lambda x, y: x * y, nums)

print("Product of all numbers:", product)
```
```
Product of all numbers: 208320
```

File Handling:

29. Load the file `about_python.txt` into your Jupyter notebook.

- Correct wrong statements about Python features.
- Count how many characters are present in the data.
- Count how many times the word ‘Python’ is repeated.
- After correcting all, print the complete valid features of Python.

In [100…
```python
with open("about_python.txt", "r") as file:
    content = file.read()

print("Original Content:\n", content)

corrected = content.replace("Python is a low-level language", "Python is a high-level language")
corrected = corrected.replace("Python does not support OOP", "Python supports Object-Oriented Programming")

char_count = len(corrected)
print("\nTotal number of characters:", char_count)

python_count = corrected.count("Python")
print("Occurrences of 'Python':", python_count)

print("\n Corrected Python Features:\n")
print(corrected)
```
```
Original Content:
 Python is a low-level language.
Python does not support OOP.
Python is an interpreted language.
Python supports multiple inheritance.


Total number of characters: 151
Occurrences of 'Python': 4

 Corrected Python Features:

Python is a high-level language.
Python supports Object-Oriented Programming.
Python is an interpreted language.
Python supports multiple inheritance.
```

In [9]:
```python
# Create a file File.txt which explains Python file handling (creating, reading, writing, appending, etc.) in detail.
# Then reload the saved file (File.txt) into your Jupyter notebook.
fl_content = """1. Creating a File:
   In Python, a file is created using the 'open()' function with the mode 'w' (write) or 'x' (exclusive creation).
   Example:
       f = open("myfile.txt", "w")

2. Writing to a File:
   Use the 'write()' method to write data into a file.
   Example:
       f.write("Hello, Python!")
       f.close()

3. Reading from a File:
   Use the 'read()', 'readline()', or 'readlines()' methods after opening the file in 'r' (read) mode.
   Example:
       f = open("myfile.txt", "r")
       print(f.read())

4. Appending to a File:
   Open the file in 'a' (append) mode to add content without deleting existing data.
   Example:
       f = open("myfile.txt", "a")
       f.write("\\nNew line added.")

5. Closing a File:
   Always close the file using 'f.close()' to free system resources.
   Example:
       f.close()"""
with open("File.txt", "w") as f:
    f.write(fl_content)

print("File 'File.txt' created successfully.")


with open("File.txt", "r") as f:
    content = f.read()

print("Content of File.txt:\n")
print(content)
```
```
File 'File.txt' created successfully.
Content of File.txt:

1. Creating a File:
   In Python, a file is created using the 'open()' function with the mode 'w' (write) or 'x' (exclusive creation).
   Example:
       f = open("myfile.txt", "w")

2. Writing to a File:
   Use the 'write()' method to write data into a file.
   Example:
       f.write("Hello, Python!")
       f.close()

3. Reading from a File:
   Use the 'read()', 'readline()', or 'readlines()' methods after opening the file in 'r' (read) mode.
   Example:
       f = open("myfile.txt", "r")
       print(f.read())

4. Appending to a File:
   Open the file in 'a' (append) mode to add content without deleting existing data.
   Example:
       f = open("myfile.txt", "a")
       f.write("\nNew line added.")

5. Closing a File:
   Always close the file using 'f.close()' to free system resources.
   Example:
       f.close()
```

In [13]:
```python
# Write a Python program that creates a text file named data.txt and add data about Python programming (at least 5 lines).
fl1_content = """Python is a low-level language.
Python does not support OOP.
Python is an interpreted language.
Python supports multiple inheritance."""
with open("data.txt", "r") as d:
    content = d.read()
```

In [14]:
```python
# Write a Python program that reads the text file named data.txt and displays its contents on the screen.
print("Content of data.txt:\n")
print(content)
```
```
Content of data.txt:

Python is a high-level language.
Python does support OOP.
Python is an interpreted language.
Python supports multiple inheritance.
Python indentation is helpful.
```

In [18]:
```python
# Write a Python program that reads a text file named data.txt and counts the total number of words in the file. Display the word count.
fl1 = "data.txt"
w = {}
with open(fl1, 'r') as file:
    for line in file:
        words = line.strip().split()
        for word in words:
            w[word] = w.get(word, 0) + 1


for word, count in w.items():
    print(f"{word}: {count}")
```
```
Python: 5
is: 3
a: 1
high-level: 1
language.: 2
does: 1
support: 1
OOP.: 1
an: 1
interpreted: 1
supports: 1
multiple: 1
inheritance.: 1
indentation: 1
helpful.: 1
```

In [22]:
```python
# Create a Python program that reads a text file named data.txt and asks the user to enter a word to search for.
# If the word is found in the file, replace all occurrences of the word with a new word provided by the user.

try:
    with open("data.txt", "r") as file:
        content = file.read()
except FileNotFoundError:
    print("File 'data.txt' not found.")
    exit()

search_word = input("Enter the word to search for: ")

if search_word in content:
    replacement_word = input("Enter the word to replace it with: ")
    updated_content = content.replace(search_word, replacement_word)

    with open("data.txt", "w") as file:
        file.write(updated_content)

    print("\n All occurrences of the word have been replaced.")
    print("\n Updated Content:\n")
    print(updated_content)
else:
    print(f"\n The word '{search_word}' was not found in the file.")
```
```
 All occurrences of the word have been replaced.

 Updated Content:

Pythonis a high-level language.
Pythondoes support OOP.
Pythonis an interpreted language.
Pythonsupports multiple inheritance.
Pythonindentation is helpful.
```

In [25]:
```python
# Write a Python program that prompts the user to enter a sentence. Append this sentence to an existing file named notes.txt.
# Make sure the newly added sentence starts on a new line.

inp_sentence = input("Enter a sentence to append to notes.txt: ")

try:
    with open("notes.txt", "a") as file:
        file.write("\n" + inp_sentence)
    print("Sentence appended to 'notes.txt' successfully.")
except Exception as e:
    print("An error occurred:", e)
```

Sentence appended to "notes.txt" successfully.