OO operations on built-in objects

Write a Python class `CustomList` that inherits from the built-in `list` class. Override the append method so that every time an item is appended, the list prints a message: `"Appending <item> to the list."`

```
class CustomList(list):
     def append(self, item):
          print(f"Appending {item} to the list.")
          super().append(item) # Call the original append method
     from the parent class
```

Add a new method to CustomList called sum_elements that returns the sum of all numeric elements in the list. If a non-numeric element is encountered, it should skip that element and continue.

```
def sum_elements(self):
     total = 0
     for item in self:
          if isinstance(item, (int, float)): # Check if the item is
     numeric
               total += item
     return total
```

Testing the modified class

# Create an instance of CustomList

```
custom_list = CustomList() # This should return an empty list
```

# Append numbers and a string

```
custom_list.append(5) # Should print: Appending 5 to the list.
custom_list.append(10) # Should print: Appending 10 to the list.
custom_list.append('hello') # Should print: Appending hello to the
list.
```

# Print the updated list

```
print("Updated List:", custom_list) # Output: Updated List: [5, 10,
'hello']
```

# Call sum_elements and print the result

```
sum_result = custom_list.sum_elements()
print("Sum of numeric elements:", sum_result) # Output: Sum of
numeric elements: 15
```