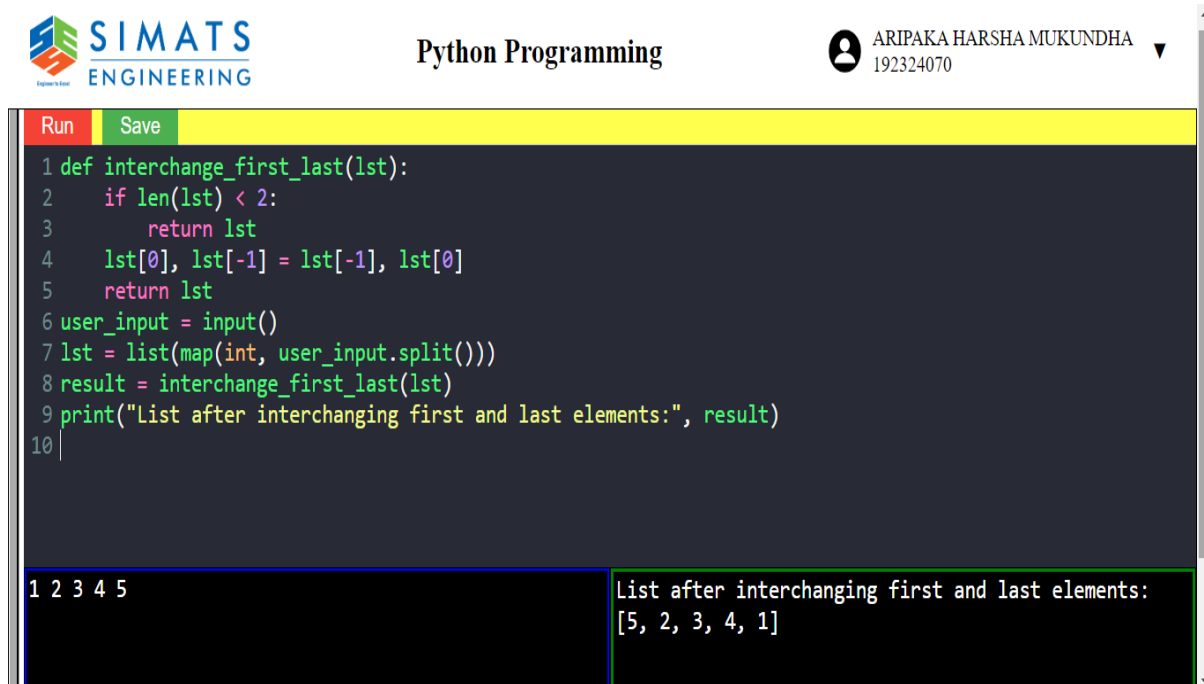


ASSIGNMENT-1

-A. Harsha Mukundha

192324070

1) Interchange First and last element in a list.

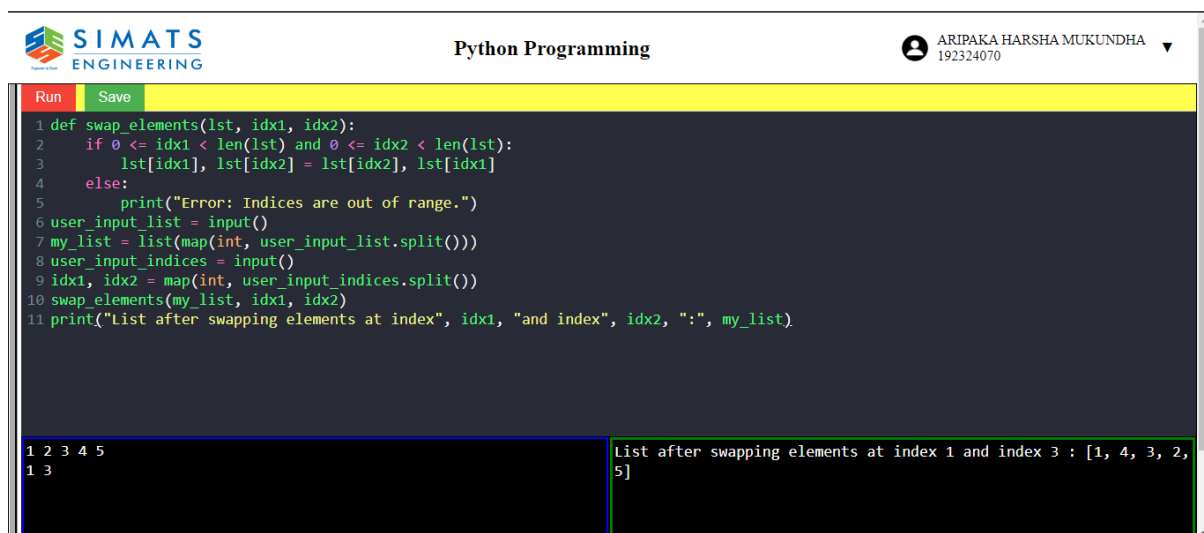


The screenshot shows a Python programming environment with the SIMATS ENGINEERING logo, the title 'Python Programming', and the user 'ARIPAKA HARSHA MUKUNDHA' with ID '192324070'. The code defines a function `interchange_first_last` that swaps the first and last elements of a list. The user inputs the list `1 2 3 4 5`, and the output is `List after interchanging first and last elements: [5, 2, 3, 4, 1]`.

```
1 def interchange_first_last(lst):
2     if len(lst) < 2:
3         return lst
4     lst[0], lst[-1] = lst[-1], lst[0]
5     return lst
6 user_input = input()
7 lst = list(map(int, user_input.split()))
8 result = interchange_first_last(lst)
9 print("List after interchanging first and last elements:", result)
10
```

1 2 3 4 5 List after interchanging first and last elements:
[5, 2, 3, 4, 1]

2) Swapping In a list

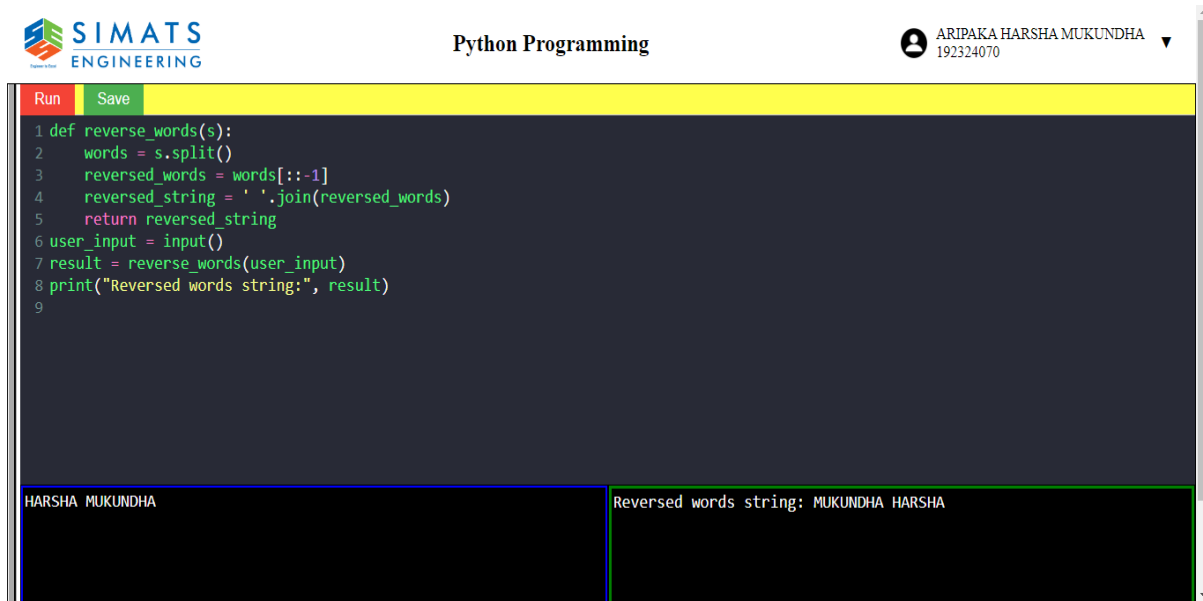


The screenshot shows a Python programming environment with the SIMATS ENGINEERING logo, the title 'Python Programming', and the user 'ARIPAKA HARSHA MUKUNDHA' with ID '192324070'. The code defines a function `swap_elements` that swaps elements at two indices in a list. The user inputs the list `1 2 3 4 5` and indices `1 3`. The output is `List after swapping elements at index 1 and index 3 : [1, 4, 3, 2, 5]`.

```
1 def swap_elements(lst, idx1, idx2):
2     if 0 <= idx1 < len(lst) and 0 <= idx2 < len(lst):
3         lst[idx1], lst[idx2] = lst[idx2], lst[idx1]
4     else:
5         print("Error: Indices are out of range.")
6 user_input_list = input()
7 my_list = list(map(int, user_input_list.split()))
8 user_input_indices = input()
9 idx1, idx2 = map(int, user_input_indices.split())
10 swap_elements(my_list, idx1, idx2)
11 print("List after swapping elements at index", idx1, "and index", idx2, ":", my_list)
```

1 2 3 4 5 List after swapping elements at index 1 and index 3 : [1, 4, 3, 2, 5]
1 3

3) Reverse a string

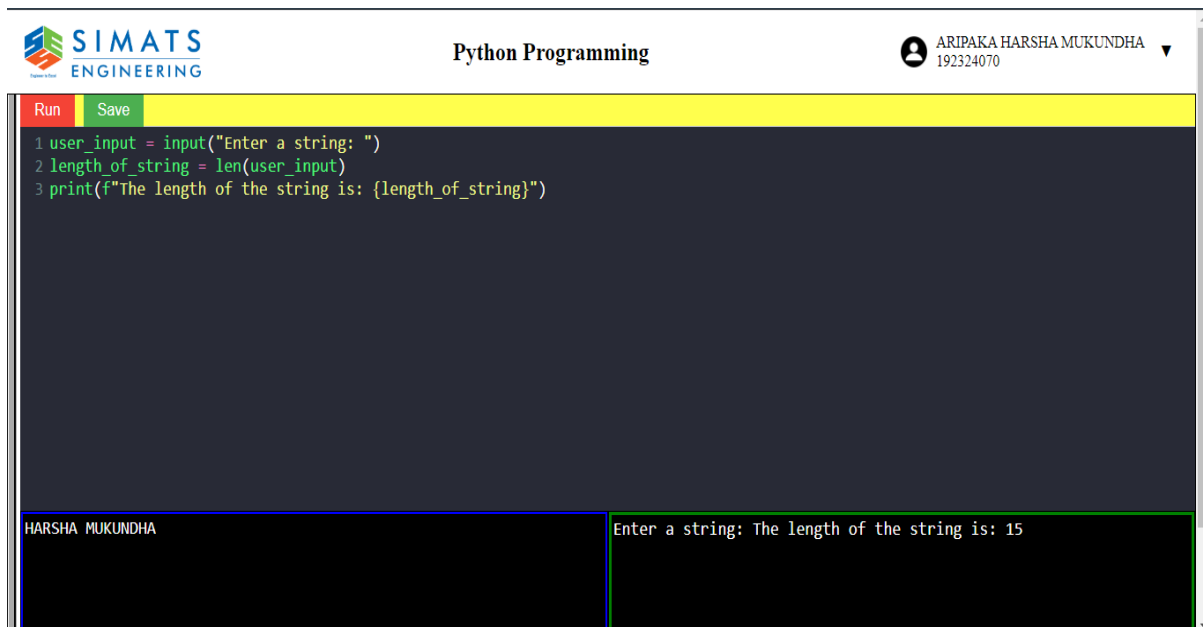


The screenshot shows a Python programming environment with the SIMATS ENGINEERING logo in the top left, the title "Python Programming" in the center, and a user profile "ARIPAKA HARSHA MUKUNDHA" with ID "192324070" in the top right. The environment has a yellow header bar with "Run" and "Save" buttons. The main area is a dark blue code editor containing the following Python code:

```
1 def reverse_words(s):  
2     words = s.split()  
3     reversed_words = words[::-1]  
4     reversed_string = ' '.join(reversed_words)  
5     return reversed_string  
6 user_input = input()  
7 result = reverse_words(user_input)  
8 print("Reversed words string:", result)  
9
```

Below the code editor is a black console area with two sections. The left section shows the input "HARSHA MUKUNDHA". The right section shows the output "Reversed words string: MUKUNDHA HARSHA".

4) length of string using slicing

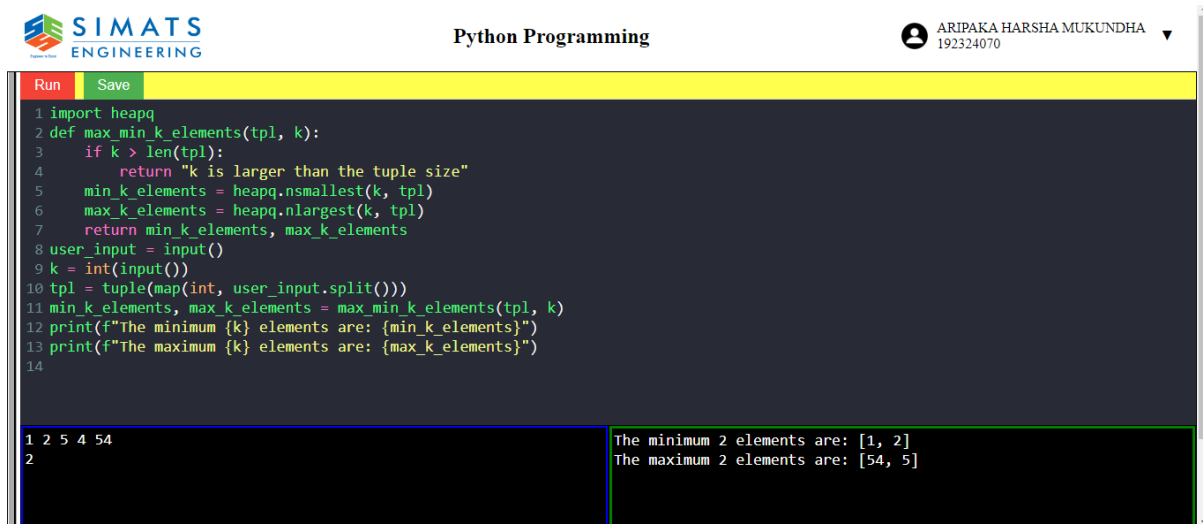


The screenshot shows a Python programming environment with the SIMATS ENGINEERING logo in the top left, the title "Python Programming" in the center, and a user profile "ARIPAKA HARSHA MUKUNDHA" with ID "192324070" in the top right. The environment has a yellow header bar with "Run" and "Save" buttons. The main area is a dark blue code editor containing the following Python code:

```
1 user_input = input("Enter a string: ")  
2 length_of_string = len(user_input)  
3 print(f"The length of the string is: {length_of_string}")
```

Below the code editor is a black console area with two sections. The left section shows the input "HARSHA MUKUNDHA". The right section shows the output "Enter a string: The length of the string is: 15".

5) Maximum and Minimum kth Elements in a list



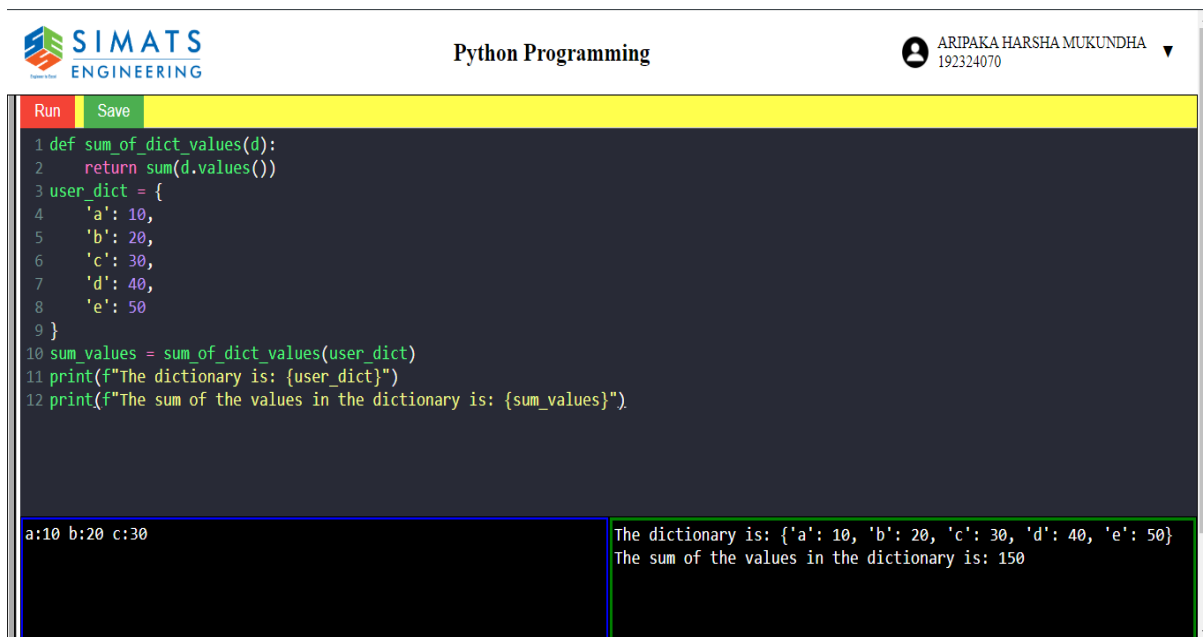
The screenshot shows a Python IDE with the SIMATS ENGINEERING logo and the title 'Python Programming'. The user is ARIPAKA HARSHA MUKUNDHA with ID 192324070. The code defines a function `max_min_k_elements` that uses `heapq.nsmallest` and `heapq.nlargest` to find the minimum and maximum k elements in a tuple. The program takes user input for a list of numbers and a value k, then prints the results.

```
1 import heapq
2 def max_min_k_elements(tpl, k):
3     if k > len(tpl):
4         return "k is larger than the tuple size"
5     min_k_elements = heapq.nsmallest(k, tpl)
6     max_k_elements = heapq.nlargest(k, tpl)
7     return min_k_elements, max_k_elements
8 user_input = input()
9 k = int(input())
10 tpl = tuple(map(int, user_input.split()))
11 min_k_elements, max_k_elements = max_min_k_elements(tpl, k)
12 print(f"The minimum {k} elements are: {min_k_elements}")
13 print(f"The maximum {k} elements are: {max_k_elements}")
14
```

Input: 1 2 5 4 54
2

Output: The minimum 2 elements are: [1, 2]
The maximum 2 elements are: [54, 5]

6) Sum of All items in a dictionary



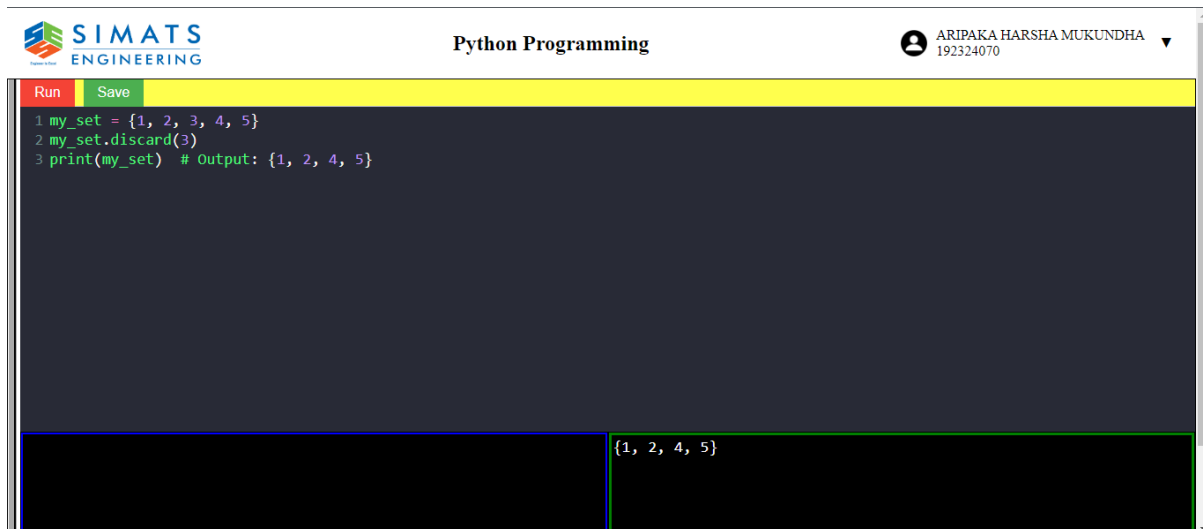
The screenshot shows a Python IDE with the SIMATS ENGINEERING logo and the title 'Python Programming'. The user is ARIPAKA HARSHA MUKUNDHA with ID 192324070. The code defines a function `sum_of_dict_values` that calculates the sum of all values in a dictionary. The program creates a dictionary with keys 'a' through 'e' and values 10 through 50, then prints the dictionary and the sum of its values.

```
1 def sum_of_dict_values(d):
2     return sum(d.values())
3 user_dict = {
4     'a': 10,
5     'b': 20,
6     'c': 30,
7     'd': 40,
8     'e': 50
9 }
10 sum_values = sum_of_dict_values(user_dict)
11 print(f"The dictionary is: {user_dict}")
12 print(f"The sum of the values in the dictionary is: {sum_values}")
```

Input: a:10 b:20 c:30

Output: The dictionary is: {'a': 10, 'b': 20, 'c': 30, 'd': 40, 'e': 50}
The sum of the values in the dictionary is: 150

7) Remove items from the set



The screenshot shows a web-based Python IDE interface. At the top, there is a header with the 'SIMATS ENGINEERING' logo on the left, the text 'Python Programming' in the center, and a user profile 'ARIPAKA HARSHA MUKUNDHA' with ID '192324070' on the right. Below the header is a yellow bar containing 'Run' and 'Save' buttons. The main area is a dark-themed code editor with the following Python code:

```
1 my_set = {1, 2, 3, 4, 5}
2 my_set.discard(3)
3 print(my_set) # Output: {1, 2, 4, 5}
```

At the bottom of the editor, there is a split view. The left pane is empty, and the right pane displays the output of the code: `{1, 2, 4, 5}`.