



main.py

```
1  #A course contains internal assessment of 50 marks and external assessment of 50 each. A st
2  def is_passed(internal_marks, external_marks):
3      if internal_marks >= 30 and external_marks > 20:
4          return True
5      else:
6          return False
7
8  internal_marks = 35
9  external_marks = 25
10 result = is_passed(internal_marks, external_marks)
11
12 if result:
13     print("The student has passed the course.")
14 else:
15     print("The student has not passed the course.")
16
```

The student has passed the course.

...Program finished with exit code 0
Press ENTER to exit console.



main.py

```
1 #course contains internal assessment of 50 marks and external assessment of 50 each. A student
2 def is_passed(internal_marks, external_marks):
3     return internal_marks >= 30 and external_marks > 20
4 students_data = {
5     "Alice": [35, 25],
6     "Bob": [28, 30],
7     "Charlie": [40, 15],
8     "David": [30, 25]
9 }
10 passed_students = []
11 for student, marks in students_data.items():
12     internal_marks, external_marks = marks[0], marks[1]
13     if is_passed(internal_marks, external_marks):
14         passed_students.append(student)
15 print("List of students who have passed the exam:")
16 for student in passed_students:
17     print(student)
18
```

List of students who have passed the exam:
Alice
David

...Program finished with exit code 0
Press ENTER to exit console.





main.py

```
1 #In a debate there are two team - positive, negative:Ten applicants ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J']
2 import random
3 applicants = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J']
4 random.shuffle(applicants)
5 positive_team = applicants[:5]
6 negative_team = applicants[5:]
7 print("Positive Team:", positive_team)
8 print("Negative Team:", negative_team)
9
```

⌵ ↶ ⚙️ 📄 input

Positive Team: ['D', 'J', 'G', 'B', 'I']
Negative Team: ['H', 'E', 'C', 'F', 'A']

...Program finished with exit code 0
Press ENTER to exit console.



main.py

```
1  #Write a function to return factorial of number using recursiondefined by lambda function
2  factorial = lambda n: 1 if n == 0 else n * factorial(n - 1)
3  num = 5
4  print("Factorial of", num, "is", factorial(num))
5
```



input

Factorial of 5 is 120

```
...Program finished with exit code 0
Press ENTER to exit console.
```





main.py

```
1  #Write a program to arrange the character of string in ascending order of their ASCII value
2  def arrange_chars_ascending(input_string):
3      chars_list = list(input_string)
4      sorted_chars = sorted(chars_list)
5      sorted_string = ''.join(sorted_chars)
6
7      return sorted_string
8  input_string = "hello"
9  result = arrange_chars_ascending(input_string)
10 print("String with characters arranged in ascending order of their ASCII values:", result)
11
```

String with characters arranged in ascending order of their ASCII values: ehllø

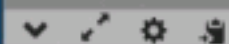
...Program finished with exit code 0
Press ENTER to exit console.





main.py

```
1 #Write a program to find difference(subtraction) between maximum and minimum element of list
2 def find_difference(lst):
3     if not lst:
4         return "List is empty"
5     max_element = max(lst)
6     min_element = min(lst)
7     difference = max_element - min_element
8     return difference
9 my_list = [5, 10, 3, 8, 15]
10 result = find_difference(my_list)
11 print("Difference between maximum and minimum elements:", result)
12
```



input

Difference between maximum and minimum elements: 12

...Program finished with exit code 0
Press ENTER to exit console.



main.py

```
1  #Write a function to take list of two-digit numbers and round-
2  def round_to_nearest_ten(numbers):
3      rounded_numbers = []
4      for num in numbers:
5          rounded_num = round(num, -1)
6          rounded_numbers.append(rounded_num)
7
8      return rounded_numbers
9  input_numbers = [23, 47, 58, 91, 36]
10 rounded_result = round_to_nearest_ten(input_numbers)
11 print("Original numbers:", input_numbers)
12 print("Rounded to nearest ten:", rounded_result)
13
```

input

```
Original numbers: [23, 47, 58, 91, 36]
Rounded to nearest ten: [20, 50, 60, 90, 40]

...Program finished with exit code 0
Press ENTER to exit console.
```