**Exp No.:1(A)**

**Date:04.09.25**

**Python Program to Calculate the Average of the Best Two Out of Three Internal Test Marks**

### **Aim:**

To write a Python program that reads the marks of three internal Mathematics tests for a student and calculates the average of the best two test scores.

### **Algorithm:**

1. **Start**
2. Prompt the user to **input marks** for Test 1, Test 2, and Test 3.
3. **Store** the three marks in a list.
4. **Sort** the list in descending order (highest marks first).
5. **Select** the first two marks (best two scores) from the sorted list.
6. **Calculate the average** of these two marks.
7. **Display** the average.
8. **End**

### 

### **Code:**

m1=float(input("Enter your marks in Internal test-I:"))

m2=float(input("Enter your marks in Internal test-II:"))

m3=float(input("Enter your marks in Internal test-III:"))

marks=[m1,m2,m3]

marks.sort(reverse=True)

best\_two=marks[:2]

avg=sum(best\_two)/2

print("The best two test marks:",best\_two)

print("Average of best two tests:",avg)

### 

### **Output:**

Enter your marks in Internal test-I:90

Enter your marks in Internal test-II:85

Enter your marks in Internal test-III:58

The best two test marks: [90.0, 85.0]

Average of best two tests: 87.5

### **Result:**

Thus the program is successfully executed and the output is obtained and verified.

### 

**Exp No.:1(B)**

**Date:04.09.25**

**Python Program to Calculate the Best Two-Test Average from Three Mini-Test Scores in a Coding Bootcamp**

### **Aim:**

To write a Python program that accepts decimal marks for three mini-tests taken by a bootcamp participant and calculates the average of the best two test scores.

### **Algorithm:**

1. **Start**
2. Prompt the user to **enter marks** for three tests (they may be decimal numbers).
3. **Store** the three marks in a list.
4. **Sort** the list in descending order using sort(reverse=True).
5. **Select the top two marks** (the first two in the sorted list).
6. **Calculate the average** of the top two marks.
7. **Display** the average rounded to two decimal places.
8. **End**

### **Code:**

test1 = float(input("Enter the marks for Test 1: "))

test2 = float(input("Enter the marks for Test 2: "))

test3 = float(input("Enter the marks for Test 3: "))

marks = [test1, test2, test3]

marks.sort(reverse=True)

average = (marks[0] + marks[1]) / 2

print(f"The average of the best two test marks is: {average:.2f}")

### **Output:**

Enter the marks for Test 1: 78.5

Enter the marks for Test 2: 91.0

Enter the marks for Test 3: 84.25

The average of the best two test marks is: 87.63

### **Result:**

Thus the program is successfully executed and the output is obtained and verified

**Exp No.:1(C)**

**Date: 04.09.2025**

# **Python Program to Calculate the Best Two-Trial Average from Three Athlete Performances**

### 

### **Aim:**

To write a Python program that records an athlete’s performance in three trials and calculates the average of the best two performances.

### 

### **Algorithm:**

1. **Start**
2. Prompt the user to enter the performance values for three trials (accept decimal input).
3. Store the values in a list.
4. Sort the list in descending order using sort(reverse=True).
5. Select the top two performances from the sorted list.
6. Calculate their average.
7. Display the average rounded to two decimal places.
8. **End**

### 

### **Program:**

trial1 = float(input("Enter performance for Trial 1: "))

trial2 = float(input("Enter performance for Trial 2: "))

trial3 = float(input("Enter performance for Trial 3: "))

performances = [trial1, trial2, trial3]

performances.sort(reverse=True)

average = (performances[0] + performances[1]) / 2

print(f"The average of the best two performances is: {average:.2f}")

### **Output:**

Enter performance for Trial 1: 12.5

Enter performance for Trial 2: 13.2

Enter performance for Trial 3: 11.8

The average of the best two performances is: 12.85

### **Result:**

Thus the program is successfully executed and the output is obtained and verified

**Exp No.:2(A)**

**Date:04.09.2025**

**Python Program to Check if a Transaction ID is a Palindrome and Count the Frequency of Each Digit**

### **Aim:**

To write a Python program that checks whether a given numeric transaction ID is a palindrome and also counts the frequency of each digit.

### **Algorithm:**

1. **Start**
2. Prompt the user to enter a transaction ID (as a number or string).
3. Convert the input to a string (to process digits easily).
4. Check if the transaction ID is a palindrome by comparing it with its reverse.
5. Initialize a dictionary (or list) to count digit frequencies.
6. Loop through each digit and update its count.
7. Display whether the ID is a palindrome.
8. Display the frequency of each digit.
9. **End**

### **Program:**

transaction\_id = input("Enter the transaction ID: ")

if transaction\_id == transaction\_id[::-1]:

print("The transaction ID is a palindrome.")

else:

print("The transaction ID is not a palindrome.")

digit\_freq = {}

for digit in transaction\_id:

if digit.isdigit():

digit\_freq[digit] = digit\_freq.get(digit, 0) + 1

print("Digit frequencies:")

for digit, count in sorted(digit\_freq.items()):

print(f"Digit {digit}: {count} time(s)")

### **Output:**

Enter the transaction ID: 12321

The transaction ID is a palindrome.

Digit frequencies:

Digit 1: 2 time(s)

Digit 2: 2 time(s)

Digit 3: 1 time(s)

### **Result:**

Thus the output is successfully executed and the output is obtained and verified.

**Exp No.:2(B)**

**Date:18.09.25**

**Python Program to Check if a Railway PNR Number is a Palindrome and Count the Frequency of Each Digit**

### **Aim:**

To write a Python program that verifies whether a given railway PNR number is a palindrome and prints the frequency of each digit in the number.

### **Algorithm:**

1. **Start**
2. Prompt the user to enter a railway PNR number (as input).
3. Convert the number to a string for easy digit processing.
4. Compare the PNR number with its reverse to check for a palindrome.
5. Initialize a dictionary to store digit frequencies.
6. Traverse each character in the PNR string:  
   * If it’s a digit, update its count in the dictionary.
7. Print whether the PNR number is a palindrome.
8. Print the frequency of each digit.
9. **End**

### **Program:**

pnr = input("Enter the railway PNR number: ")

if pnr == pnr[::-1]:

print("The PNR number is a palindrome.")

else:

print("The PNR number is not a palindrome.")

digit\_freq = {}

for char in pnr:

if char.isdigit():

digit\_freq[char] = digit\_freq.get(char, 0) + 1

print("Digit frequencies:")

for digit, count in sorted(digit\_freq.items()):

print(f"Digit {digit}: {count} time(s)")

### **Output:**

Enter the railway PNR number: 1234321

The PNR number is a palindrome.

Digit frequencies:

Digit 1: 2 time(s)

Digit 2: 2 time(s)

Digit 3: 2 time(s)

Digit 4: 1 time(s)

### **Result:**

Thus the program is successfully executed and the output is obtained and verified

**Exp No.:2(C)**

**Date:18.09.25**

**Python Program to Check if a School Roll Number is a Palindrome and Count the Frequency of Each Digit**

### **Aim:**

To write a Python program that checks whether a given school roll number is a palindrome and prints how many times each digit appears in the number.

### **Algorithm:**

1. **Start**
2. Prompt the user to enter a school roll number.
3. Convert the input to a string to allow reversal and digit analysis.
4. Compare the roll number with its reverse to check for palindrome.
5. Initialize an empty dictionary to track digit frequencies.
6. Loop through each character in the roll number:  
   * If it's a digit, increment its count in the dictionary.
7. Display whether the roll number is a palindrome.
8. Display how many times each digit appears.
9. **End**

### **Program:**

roll\_number = input("Enter the school roll number: ")

if roll\_number == roll\_number[::-1]:

print("The roll number is a palindrome.")

else:

print("The roll number is not a palindrome.")

digit\_freq = {}

for char in roll\_number:

if char.isdigit():

digit\_freq[char] = digit\_freq.get(char, 0) + 1

print("Digit frequencies:")

for digit, count in sorted(digit\_freq.items()):

print(f"Digit {digit}: {count} time(s)")

### **Output:**

Enter the school roll number: 1221

The roll number is a palindrome.

Digit frequencies:

Digit 1: 2 time(s)

Digit 2: 2 time(s)

### **Result:**

Thus the program is successfully executed and the output is obtained and verified