1.1.1. Calculate Momentum

02:09

Write a program that accepts the mass of an object (in kilograms) and its velocity (in meters per second), then calculates and displays the momentum of the object. The momentum p is calculated using the formula:

$$p = m \times v$$

where:
m is the mass of the object (in kilograms).
v is the velocity of the object (in meters per second).

Input Format:
A single floating-point number representing the mass of the object in kilograms.
A single floating-point number representing the velocity of the object in meters per second.

Output Format:
The output will display calculated momentum with appropriate units (kgm/s) (rounded up to 2 decimal places).

Sample Test Cases +

Explorer

calculateM...

Submit

Debugger

```python
# Accept mass and velocity as input
mass = float(input())  # Input mass in kilograms
velocity = float(input())  # Input velocity in meters per second

# Calculate momentum
momentum = mass * velocity

# Display the result rounded to 2 decimal places
print(f"{momentum:.2f}kgm/s")

```

Average time
0.033 s
33.25 ms

Maximum time
0.047 s
47.00 ms

2 out of 2 shown test case(s) passed

2 out of 2 hidden test case(s) passed

Test case 1 32 ms Debug

Expected output	Actual output
5.0	5.0
10.0	10.0
50.00kgm/s	50.00kgm/s

< Prev Reset Submit Next >

1.1.2. Conditional Calculation Based on the ...

03:40

Write a Python program that accepts an integer n as input. Depending on the number of digits in n.

Constraints:
$1 \le n \le 999$

Input Format:
The input consists of a single integer n.

Output Format:
If n is a single-digit number, print its square.
If n is a two-digit number, print its square root (rounded to two decimal places).
If n is a three-digit number, print its cube root (rounded to two decimal places).
Else print "Invalid".

Sample Test Cases +

Explorer

conditional...

Submit

Debugger

```python
import math

# Accepting input from the user
num = int(input())

# Check the number of digits in the input
if 1 <= num <= 9:  # Single-digit number
    print(num ** 2)
elif 10 <= num <= 99:  # Two-digit number
    print(f"{math.sqrt(num):.2f}")
elif 100 <= num <= 999:  # Three-digit number
    print(f"{math.pow(num, 1/3):.2f}")
else:  # If number is out of the given range
    print("Invalid")
```

Average time
0.011 s
11.43 ms

Maximum time
0.016 s
16.00 ms

4 out of 4 shown test case(s) passed

3 out of 3 hidden test case(s) passed

Test case 1 10 ms Debug

Expected output	Actual output
9	9
81	81

1.1.3. Age and Salary Calculation

14:56

Write a Python program that reads the birth date and salary of employees.

Input Format:
The input consists of:
A string representing the birth date of the employee in the format $DD-MM-YYYY$.
A floating-point number representing the salary of the employee in rupees.

Output Format:
The output should include:
The age of the employee.
The salary of the employee in dollars.

Note:
1INR=0.012USD

Sample Test Cases +

Explorer

birthDatea...

Submit

Debugger

```python
# import the required package
from datetime import datetime
def calculate_age(birthdate):
    # Write your code here
    date_object = datetime.strptime(birthdate, "%d-%m-%Y")
    today = datetime.today()
    if ((today.month, today.day) < (date_object.month, date_object.day)):
        age = today.year-date_object.year-((today.month, today.day) < (date_object.month, date_object.day))
        return age
    elif((today.month,today.day) > (date_object.month,date_object.day)):
        age = today.year-date_object.year-((today.month, today.day) > (date_object.month, date_object.day))
    return age



def convert_salary_to_dollars(salary_in_rupees):
    # write your code here
    salary=salary_in_rupees*0.012
    return salary

birthdate = input()
salary_in_rupees = float(input())
age = calculate_age(birthdate)
salary_in_dollars = convert_salary_to_dollars(salary_in_rupees)
print(f"Age: {age}")
print(f"Salary in dollars: {salary_in_dollars:.2f}")
```

Average time
0.071 s
71.00 ms

Maximum time
0.144 s
144.00 ms

2 out of 2 shown test case(s) passed

1 out of 2 hidden test case(s) passed

Test case 1 144 ms Debug

Expected output	Actual output
15-06-1991	15-06-1991
50000	50000
Age: 33	Age: 33

< Prev Reset Submit Next >

1.1.4. Reverse a Number

15:22

You are given an integer number. Your task is to reverse the digits of the number and print the reversed number.

Input Format
The input is an integer.

Output Format
Print a single integer which is the reversed number.

Sample Test Cases +

Explorer

reverseNu...

Submit

Debugger

```python
# Accept the input as an integer
num = int(input())

# Reverse the digits by converting the number to a string, reversing it, and converting it back to an integer
reversed_num = int(str(num)[::-1])

# Output the reversed number
print(reversed_num)
```

Average time	Maximum time
0.013 s	0.017 s
12.80 ms	17.00 ms

2 out of 2 shown test case(s) passed

3 out of 3 hidden test case(s) passed

Test case 1 17 ms Debug

Expected output	Actual output
5367	5367
7635	7635

1.1.5. Multiplication Table

00:28

Write a Python program that takes an integer as input and prints the multiplication table for that integer from 1 to 10.

Input Format:
The first line of input contains an integer that represents the number for which the multiplication table is to be printed.

Output Format:
Print the multiplication table for the given number .

Sample Test Cases +

Explorer

multiplicat...

Submit

Debugger

```python
# Accept an integer input
num = int(input())

# Print the multiplication table from 1 to 10
for i in range(1, 11):
    print(f"{num} x {i} = {num * i}")

```

Average time
0.015 s
14.75 ms

Maximum time
0.017 s
17.00 ms

2 out of 2 shown test case(s) passed

2 out of 2 hidden test case(s) passed

Test case 1 17 ms Debug

Expected output	Actual output
8	8
8 x 1 = 8	8 x 1 = 8
8 x 2 = 16	8 x 2 = 16

1.2.1. Pass or Fail

33:48

Write a Python program that accepts the number of courses and the marks of a student in those courses.
The grade is determined based on the aggregate percentage:

- If the aggregate percentage is greater than 75, the grade is Distinction.
- If the aggregate percentage is greater than or equal to 60 but less than 75, the grade is First Division.
- If the aggregate percentage is greater than or equal to 50 but less than 60, the grade is Second Division.
- If the aggregate percentage is greater than or equal to 40 but less than 50, the grade is Third Division.

Input Format:
The first input will be an integer n, the number of courses.
The second input will be n integers representing the marks of the student in each of the n courses, separated by a space.

Output Format:
If the student passes all courses:

- Print the aggregate percentage (rounded to two decimal places).
- Print the grade based on the aggregate percentage.

If the student fails any course (marks < 40 in any course), print:

- "Fail".

Sample Test Cases +

Explorer

passorFail....

Submit

Debugger

```python
def calculate_grade():
    # Accept number of courses
    num_courses = int(input())

    # Accept marks for the courses
    marks = list(map(int, input().split()))

    # Check if the student has failed any course (marks < 40)
    if any(mark < 40 for mark in marks):
        print("Fail")
    else:
        # Calculate the total marks and aggregate percentage
        total_marks = sum(marks)
        aggregate_percentage = (total_marks / (num_courses * 100)) * 100

        # Print the aggregate percentage
        print(f"Aggregate Percentage: {aggregate_percentage:.2f}")

        # Determine the grade based on the aggregate percentage
        if aggregate_percentage > 75:
            print("Grade: Distinction")
        elif 60 <= aggregate_percentage < 75:
            print("Grade: First Division")
        elif 50 <= aggregate_percentage < 60:
            print("Grade: Second Division")
        elif 40 <= aggregate_percentage < 50:
            print("Grade: Third Division")

# Call the function to execute
calculate_grade()
```

Average time
0.023 s
22.75 ms

Maximum time
0.027 s
27.00 ms

2 out of 2 shown test case(s) passed

2 out of 2 hidden test case(s) passed

Test case 1 27 ms Debug

Expected output	Actual output
5	5
56 78 97 86 93	56 78 97 86 93
Aggregate Percentage: 82.00	Aggregate Percentage: 82.00

1.2.2. Fibonacci series using Recursive Func... 14:07

Write a Python program to find the Fibonacci series of a given number of terms using recursive function calls.

Expected Output-1:
Enter terms for Fibonacci series: 5
0 1 1 2 3

Expected Output-2:
Enter terms for Fibonacci series: 9
0 1 1 2 3 5 8 13 21

Instructions:

- Your input and output must follow the input and output layout mentioned in the visible sample test case.
- Hidden test cases will only pass when users' input and output match the expected input and output.

Sample Test Cases +

Explorer

fib.py

Submit

```
1  def fib(n,a=0,b=1):
2      if n==0:
3          return a
4      else:
5          return fib(n-1,b,a+b)
6
7  n=int(input("Enter terms for
   Fibonacci series: "))
8  for i in range (n):
9      print(fib(i),end=" ")
```

Debugger

Average time
0.014 s
14.25 ms

Maximum time
0.015 s
15.00 ms

2 out of 2 shown test case(s) passed

2 out of 2 hidden test case(s) passed

Test case 1 15 ms Debug

Expected output	Actual output
Enter terms for Fibonacci series: 5	Enter terms for Fibonacci series: 5
0 1 1 2 3	0 1 1 2 3

1.2.3. Pattern - 1

02:27

Write a Python program to print a pattern of asterisks in the form of a right-angled triangle.

Input Format:
The input is an integer, representing the number of rows in the pattern.

Output Format
The output should display the pattern of asterisks (*), with each row containing an increasing number of asterisks.

Note:
Refer to the displayed test cases for the sample pattern.

Sample Test Cases +

Explorer

rightangle...

Submit

Debugger

```
1  n=int(input())
2  for i in range(1, n + 1):
3      print('* ' * i)
4
5
```

Average time
0.013 s
13.00 ms

Maximum time
0.017 s
17.00 ms

2 out of 2 shown test case(s) passed

4 out of 4 hidden test case(s) passed

Test case 1 17 ms

Debug

Expected output	Actual output
5	5
*	*
* *	* *

1.2.4. Pattern - 2

09:39

Write a Python program to print a right-angled triangle pattern of numbers.

Input Format:
The input is an integer, representing the number of rows in the pattern.

Output Format:
The output should display the pattern of numbers, with each row containing increasing numbers starting from 1 up to the row number.

Note:
Refer to the displayed test cases for the sample pattern.

Sample Test Cases +

Explorer

numberPat...

Submit

Debugger

```python
# Accept an integer input for the number of rows
n = int(input())

# Loop to print the pattern
for i in range(1, n + 1):
    for j in range(1, i + 1):
        print(j, end=" ")
    print()  # Move to the next line after each row

```

Average time
0.015 s
14.75 ms

Maximum time
0.017 s
17.00 ms

2 out of 2 shown test case(s) passed

2 out of 2 hidden test case(s) passed

Test case 1 17 ms Debug

Expected output	Actual output
5	5
1	1
1 2	1 2