Exercises - Batch 1

Define the functions required to solve each exercise and test them by running the example code.

1. Basic Function Practice

Write a function that takes two numbers and returns their sum, difference, product, and quotient as a tuple.

Example:

```
print(basic_operations(8, 2))
# Output: (10, 6, 16, 4.0)
```

2. Area and Perimeter of a Rectangle

Define a function `rectangle_properties` that takes the length and width of a rectangle and returns its area and perimeter.

Hint: Use the formula for the area: $\text{Area} = \text{length} \times \text{width}$, and for the perimeter: $\text{Perimeter} = 2 \times (\text{length} + \text{width})$.

Example:

```
area, perimeter = rectangle_properties(5, 3))
assert area == 15
assert perimeter == 16
```

3. Simple Conversion (Celsius to Fahrenheit)

Write a function `celsius_to_fahrenheit` that converts a temperature in Celsius to Fahrenheit using the formula: $F = C \times \frac{9}{5} + 32$.

Example:

```
# Test with:
print(celsius_to_fahrenheit(25))
# Output: 77.0
```

4. Pythagorean Theorem

Write a function `hypotenuse` that calculates the hypotenuse of a right triangle given the two other sides a and b, using the Pythagorean theorem: $c = \sqrt{a^2 + b^2}$. You can `import` the `math` library to solve the exercise.

```
assert hypotenuse(3, 4) == 5
```

5. Distance Formula

Define a function `distance_between_points` that calculates the distance between two points in 2D space, given their coordinates (x_1, y_1) and (x_2, y_2). Use the distance formula:
$\text{distance} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$.

Example:

```
# Test with:
print(distance_between_points(0, 0, 3, 4))
# Output: 5.0
```

6. Average of Three Numbers

Write a function `average_of_three` that takes three numbers and returns their average.

Example:

```
avg = average_of_three(1, 2, 3)
avg == 2.0 # True
```

7. Volume of a Cylinder

Define a function `cylinder_volume` that calculates the volume of a cylinder, given the radius (r) and height (h). Use the formula: $V = \pi r^2 h$.

Example:

```
print(cylinder_volume(3, 5))
# Output: 141.3716694115407
```

Here are more advanced exercises that still avoid using conditionals and loops but involve more complex mathematical concepts or require combining multiple operations.

8. Quadratic Equation Solver

Write a function `quadratic_solver` that calculates the roots of a quadratic equation $ax^2 + bx + c = 0$ using the quadratic formula: $x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$.
Return both possible roots as a tuple.

Note: Ignore complex roots for now (assume the discriminant $b^2 - 4ac$ is positive).

Example:

```
assert quadratic_solver(1, -3, 2) == (2.0 , 1.0)
```

9. Dot Product of Two Vectors

Write a function `dot_product` that calculates the dot product of two vectors in 3D space, where each vector is represented by its coordinates `v1 = [x1, y1, z1]` and `v2 = [x2, y2, z2]`. The result of a dot product is a scalar given by the formula

$$\mathbf{v}_1 \cdot \mathbf{v}_2 = x_1x_2 + y_1y_2 + z_1z_2$$

Example:

```
v1 = [1, 2, 3]
v2 = [4, 5, 6]
assert dot_product(v1, v2) == 32
```

10. Cross Product of Two Vectors

Write a function `cross_product` that calculates the cross product of two vectors in 3D space, where each vector is represented by its coordinates `v1 = [x1, y1, z1]` and `v2 = [x2, y2, z2]`. The result of cross product is a vector (a list in Python) given by

$$\mathbf{v}_1 \times \mathbf{v}_2 = (y_1z_2 - z_1y_2, z_1x_2 - x_1z_2, x_1y_2 - y_1x_2)$$

Example:

```
v1 = [1, 2, 3]
v2 = [4, 5, 6]
vcross = cross_product(v1, v2)
assert vcross == [-3, 6, -3]
```

11. Determinant of a 2x2 Matrix

Write a function `determinant_2x2` that takes a 2x2 matrix (a list of lists) and returns its determinant. The determinant of a matrix

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \tag{1}$$

is a scalar given by $\det(A) = ad - bc$.

Example:

```
A = [[1, 2],   # We represent a matrix as a list whose elements are lists themeselves.
     [3, 4]]   # A matrix can be seen as a list of rows.

determinant_2x2(A) # == -2
```

12. Matrix Addition (2x2)

Exercise: Write a function `matrix_addition_2x2` that takes two 2x2 matrices and returns their sum. The sum of two matrices is a matrix whose elements are the sum of the corrisponding elements in the 2 input matrices.

Assume that the input matrices are given as list of lists and that the output should be another list of lists.

Example:

```
A = [[1, 2], [3, 4]]
B = [[5, 6], [7, 8]]
assert matrix_addition_2x2(A, B) == [[6, 8], [10, 12]]
```