# **TASK 2 – MATHS LIBRARY**

## math.ceil(*x*)

The math.ceil() method rounds a number UP to the nearest integer, if necessary, and returns the result.

## math.comb(*n*, *k*)

The math.comb() method returns the number of ways of picking k unordered outcomes from n possibilities, without repetition, also known as combinations.

## math.copysign(*x*, *y*)

The math.copysign() method returns a float consisting of the value of the first parameter and the sign(+/-) of the second parameter.

math.**fabs**(*x*)

The math.fabs() method returns the absolute value of a number, as a float. Absolute denotes a non-negative number. This removes the negative sign of the value if it has any. Unlike Python [abs()](https://www.w3schools.com/python/ref_func_abs.asp), this method always converts the value to a float value.

## math.factorial(*n*)

Return n factorial as an integer. Raises [ValueError](https://docs.python.org/3/library/exceptions.html" \l "ValueError" \o "ValueError) if n is not integral or is negative.

## math.floor(*x*)

The math.floor() method rounds a number DOWN to the nearest integer, if necessary, and returns the result.

## math.fmod(*x*, *y*)

The math.fmod() method returns the remainder (modulo) of x/y. If both x and y = 0, it returns a ValueError. If y = 0, it returns a ValueError. If x or y is not a number, it returns a TypeError.

## math.frexp(*x*)

The math.frexp() method returns the mantissa and the exponent of a specified number, as a pair (m,e). The mathematical formula for this method is: number = m \* 2\*\*e.

## math.fsum(*iterable*)

The math.fsum() method returns the sum of all items in any iterable (tuples, arrays, lists, etc.).

## math.cbrt(*x*)

Return the cube root of x.

## math.exp(*x*)

Return e raised to the power x, where e = 2.718281… is the base of natural logarithms. This is usually more accurate than math.e \*\* x or pow(math.e, x).

## math.exp2(*x*)

Return 2 raised to the power x.

## math.expm1(*x*)

Return e raised to the power x, minus 1. Here e is the base of natural logarithms. For small floats x, the subtraction in exp(x) - 1 can result in a [significant loss of precision](https://en.wikipedia.org/wiki/Loss_of_significance); the [expm1()](https://docs.python.org/3/library/math.html" \l "math.expm1" \o "math.expm1) function provides a way to compute this quantity to full precision.

## math.log10(*x*)

Return the base-10 logarithm of x. This is usually more accurate than log(x, 10).

## math.pow(*x*, *y*)

Return x raised to the power y. Exceptional cases follow the IEEE 754 standard as far as possible. In particular, pow(1.0, x) and pow(x, 0.0) always return 1.0, even when x is a zero or a NaN. If both x and y are finite, x is negative, and y is not an integer then pow(x, y) is undefined, and raises [ValueError](https://docs.python.org/3/library/exceptions.html" \l "ValueError" \o "ValueError).

## math.sqrt(*x*)

Return the square root of x.

## math.isfinite(*x*)

Return True if x is neither an infinity nor a NaN, and False otherwise. (Note that 0.0 is considered finite.)

## math.isinf(*x*)

Return True if x is a positive or negative infinity, and False otherwise.

## math.isnan(*x*)

Return True if x is a NaN (not a number), and False otherwise.

## math.isqrt(*n*)

Return the integer square root of the nonnegative integer n. This is the floor of the exact square root of n, or equivalently the greatest integer a such that a² ≤ n.

## math.lcm(*\*integers*)

Return the least common multiple of the specified integer arguments. If all arguments are nonzero, then the returned value is the smallest positive integer that is a multiple of all arguments. If any of the arguments is zero, then the returned value is 0. lcm() without arguments returns 1.

## math.ldexp(*x*, *i*)

Return x \* (2\*\*i). This is essentially the inverse of function [frexp()](https://docs.python.org/3/library/math.html" \l "math.frexp" \o "math.frexp).

## math.modf(*x*)

Return the fractional and integer parts of x. Both results carry the sign of x and are floats.

## math.perm(*n*, *k=None*)

Return the number of ways to choose k items from n items without repetition and with order.

## math.prod(iterable, \*, start=1)

Calculate the product of all the elements in the input iterable. The default start value for the product is 1. When the iterable is empty, return the start value. This function is intended specifically for use with numeric values and may reject non-numeric types.

# **TASK 3 - NUMPY**

## data.shape()

In NumPy we will use an attribute called shape which returns a tuple, the elements of the tuple give the lengths of the corresponding array dimensions.

## arr.dtype()

The NumPy array object has a property called dtype that returns the data type of the array

## ndim()

ndim() function return the number of dimensions of an array.

## numpy.zeros()

The numpy.zeros() function returns a new array of given shape and type, with zeros.

## numpy.empty()

Return a new array of given shape and type, without initializing entries.

## numpy.sqrt()

Return the non-negative square-root of an array, element-wise.

## numpy.exp()

Calculate the exponential of all elements in the input array.

## random.randint()

The randint() method returns an integer number selected element from the specified range.

## random.choice()

The choice() method returns a randomly selected element from the specified sequence. The sequence can be a string, a range, a list, a tuple or any other kind of sequence.

## numpy.maximum()

Element-wise maximum of array elements. Compare two arrays and returns a new array containing the element-wise maxima. If one of the elements being compared is a NaN, then that element is returned. If both elements are NaNs then the first is returned. The latter distinction is important for complex NaNs, which are defined as at least one of the real or imaginary parts being a NaN. The net effect is that NaNs are propagated.

## numpy.add()

Add arguments element-wise.

## numpy.mean()

Compute the arithmetic mean along the specified axis. Returns the average of the array elements. The average is taken over the flattened array by default, otherwise over the specified axis. [float64](https://numpy.org/doc/stable/reference/arrays.scalars.html" \l "numpy.float64" \o "numpy.float64) intermediate and return values are used for integer inputs.

## numpy.sum()

Sum of array elements over a given axis.

## numpy.copy()

Return an array copy of the given object.

## numpy.ndarray.view()

New view of array with the same data.

## numpy.reshape()

Gives a new shape to an array without changing its data.

## numpy.cumsum()

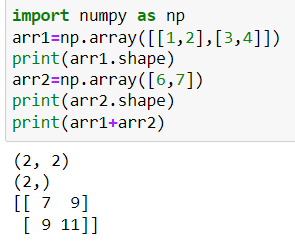
Return the cumulative sum of the elements along a given axis.

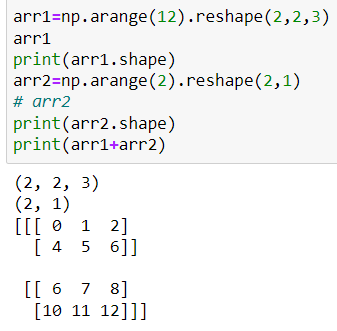
## numpy.sort()

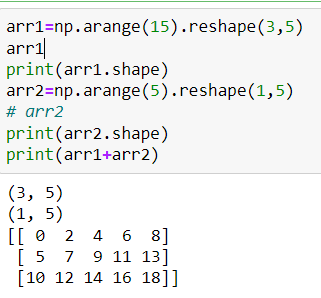
Return a sorted copy of an array.

# BROADCASTING EXAMPLES

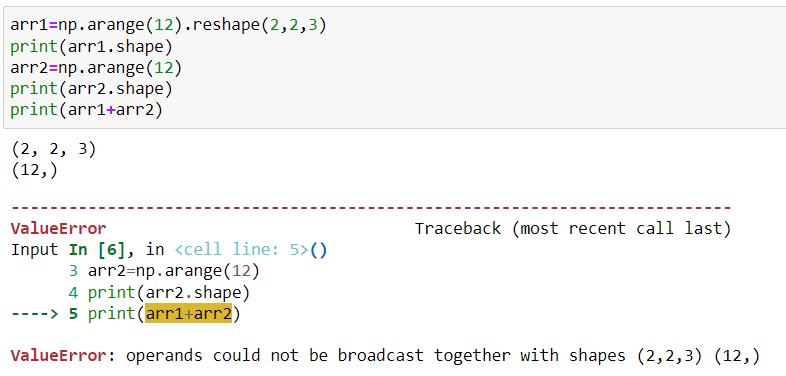
SUCCESS CASES



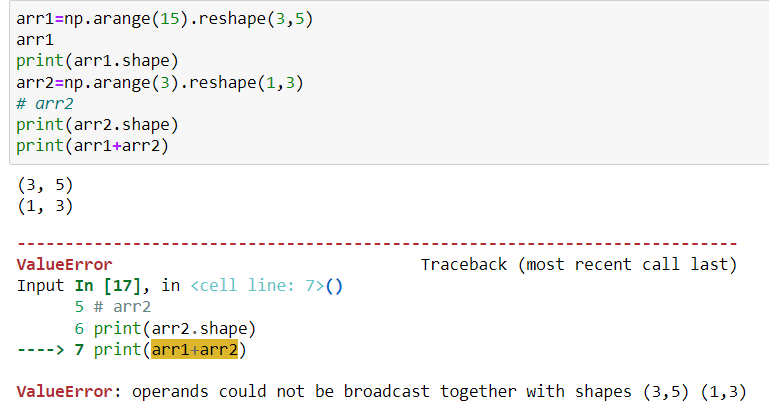




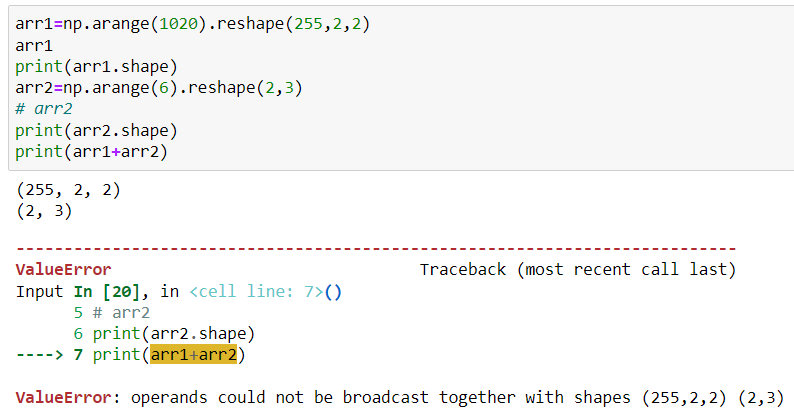
FAILED CASES



In the above example, the reason of failure is the number of rows does not have common dimension.



Cannot be broadcasted because the number of columns is 5 in array 1 and 3 in array 2. It can be broadcasted if we take transpose of the array 2.



In this example, one dimension of the array matches with other while other does not. In array 2, if we replace 3 with 2 or 1, then it can be broadcasted.