**Functions for Rounding –**  
[**numpy.around(arr, decimals = 0, out = None)**](https://www.geeksforgeeks.org/numpy-around-python/)**:**This mathematical function helps user to evenly round array elements to the given number of decimals.

|  |
| --- |
| # Python program explaining  # around() function    import numpy as np    in\_array = [.5, 1.5, 2.5, 3.5, 4.5, 10.1]  print ("Input array : \n", in\_array)    round\_off\_values = np.around(in\_array)  print ("\nRounded values : \n", round\_off\_values)      in\_array = [.53, 1.54, .71]  print ("\nInput array : \n", in\_array)    round\_off\_values = np.around(in\_array)  print ("\nRounded values : \n", round\_off\_values)    in\_array = [.5538, 1.33354, .71445]  print ("\nInput array : \n", in\_array)    round\_off\_values = np.around(in\_array, decimals = 3)  print ("\nRounded values : \n", round\_off\_values) |

Run on IDE

**Output :**

Input array :

[0.5, 1.5, 2.5, 3.5, 4.5, 10.1]

Rounded values :

[ 0. 2. 2. 4. 4. 10.]

Input array :

[0.53, 1.54, 0.71]

Rounded values :

[ 1. 2. 1.]

Input array :

[0.5538, 1.33354, 0.71445]

Rounded values :

[ 0.554 1.334 0.714]

[**numpy.round\_(arr, decimals = 0, out = None)**](https://www.geeksforgeeks.org/numpy-round_-python/)**:**This mathematical function round an array to the given number of decimals.

|  |
| --- |
| # Python program explaining  # round\_() function  import numpy as np    in\_array = [.5, 1.5, 2.5, 3.5, 4.5, 10.1]  print ("Input array : \n", in\_array)    round\_off\_values = np.round\_(in\_array)  print ("\nRounded values : \n", round\_off\_values)      in\_array = [.53, 1.54, .71]  print ("\nInput array : \n", in\_array)    round\_off\_values = np.round\_(in\_array)  print ("\nRounded values : \n", round\_off\_values)    in\_array = [.5538, 1.33354, .71445]  print ("\nInput array : \n", in\_array)    round\_off\_values = np.round\_(in\_array, decimals = 3)  print ("\nRounded values : \n", round\_off\_values) |

Run on IDE

**Output :**

Input array :

[0.5, 1.5, 2.5, 3.5, 4.5, 10.1]

Rounded values :

[ 0. 2. 2. 4. 4. 10.]

Input array :

[0.53, 1.54, 0.71]

Rounded values :

[ 1. 2. 1.]

Input array :

[0.5538, 1.33354, 0.71445]

Rounded values :

[ 0.554 1.334 0.714]

|  |  |
| --- | --- |
| **FUNCTION** | **DESCRIPTION** |
| [**rint()**](https://www.geeksforgeeks.org/numpy-rint-python/) | Round to nearest integer towards zero. |
| [**fix()**](https://www.geeksforgeeks.org/numpy-fix-python/) | Round to nearest integer towards zero. |
| [**floor()**](https://www.geeksforgeeks.org/numpy-arccos-python/) | Return the floor of the input, element-wise. |
| [**ceil()**](https://www.geeksforgeeks.org/numpy-ceil-python/) | Return the ceiling of the input, element-wise. |
| [**trunc()**](https://www.geeksforgeeks.org/numpy-trunc-python/) | Return the truncated value of the input, element-wise. |

**Exponents and logarithms Functions –**  
[**numpy.exp(array, out = None, where = True, casting = ‘same\_kind’, order = ‘K’, dtype = None)**](https://www.geeksforgeeks.org/numpy-exp-python/)**:**This mathematical function helps user to calculate exponential of all the elements in the input array.

|  |
| --- |
| # Python program explaining  # exp() function  import numpy as np    in\_array = [1, 3, 5]  print ("Input array : ", in\_array)    out\_array = np.exp(in\_array)  print ("Output array : ", out\_array) |

Run on IDE

**Output :**

Input array : [1, 3, 5]

Output array : [ 2.71828183 20.08553692 148.4131591 ]

[**numpy.log(x[, out] = ufunc ‘log1p’)**](https://www.geeksforgeeks.org/numpy-log-python/)**:**This mathematical function helps user to calculate **Natural logarithm of x** where x belongs to all the input array elements.  
Natural logarithm log is the **inverse of the exp()**, so that log(exp(x)) = x. The natural logarithm is log in base e.

|  |
| --- |
| # Python program explaining  # log() function  import numpy as np    in\_array = [1, 3, 5, 2\*\*8]  print ("Input array : ", in\_array)    out\_array = np.log(in\_array)  print ("Output array : ", out\_array)      print("\nnp.log(4\*\*4) : ", np.log(4\*\*4))  print("np.log(2\*\*8) : ", np.log(2\*\*8)) |

Run on IDE

**Output :**

Input array : [1, 3, 5, 256]

Output array : [ 0. 1.09861229 1.60943791 5.54517744]

np.log(4\*\*4) : 5.54517744448

np.log(2\*\*8) : 5.54517744448

|  |  |
| --- | --- |
| **FUNCTION** | **DESCRIPTION** |
| [**expm1()**](https://www.geeksforgeeks.org/numpy-expm1-python/) | Calculate exp(x) – 1 for all elements in the array. |
| [**exp2()**](https://www.geeksforgeeks.org/numpy-exp2-python/) | Calculate 2\*\*p for all p in the input array. |
| [**log10()**](https://www.geeksforgeeks.org/numpy-log10-python/) | Return the base 10 logarithm of the input array, element-wise. |
| [**log2()**](https://www.geeksforgeeks.org/numpy-log2-python/) | Base-2 logarithm of x. |
| [**log1p()**](https://www.geeksforgeeks.org/numpy-log1p-python/) | Return the natural logarithm of one plus the input array, element-wise. |
| [**logaddexp()**](https://www.geeksforgeeks.org/numpy-logaddexp-in-python/) | Logarithm of the sum of exponentiations of the inputs. |
| [**logaddexp2()**](https://www.geeksforgeeks.org/numpy-logaddexp2-in-python/) | Logarithm of the sum of exponentiations of the inputs in base-2. |

**Arithmetic Functions –**  
[**numpy.reciprocal(x, /, out=None, \*, where=True)**](https://www.geeksforgeeks.org/numpy-reciprocal-in-python/)**:**This mathematical function is used to calculate reciprocal of all the elements in the input array.

**Note:**For integer arguments with absolute value larger than 1, the result is always zero because of the way Python handles integer division. For integer zero the result is an overflow.

|  |
| --- |
| # Python3 code demonstrate reciprocal() function    # importing numpy  import numpy as np    in\_num = 2.0  print ("Input  number : ", in\_num)    out\_num = np.reciprocal(in\_num)  print ("Output number : ", out\_num) |

Run on IDE

**Output :**

Input number : 2.0

Output number : 0.5

[**numpy.divide(arr1, arr2, out = None, where = True, casting = ‘same\_kind’, order = ‘K’, dtype = None)**](https://www.geeksforgeeks.org/numpy-divide-python/)**:**Array element from first array is divided by elements from second element (all happens element-wise). Both arr1 and arr2 must have same shape and element in arr2 must not be zero; otherwise it will raise an error.

|  |
| --- |
| # Python program explaining  # divide() function  import numpy as np    # input\_array  arr1 = [2, 27, 2, 21, 23]  arr2 = [2, 3, 4, 5, 6]  print ("arr1         : ", arr1)  print ("arr2         : ", arr2)    # output\_array  out = np.divide(arr1, arr2)  print ("\nOutput array : \n", out) |

Run on IDE

**Output :**

arr1 : [2, 27, 2, 21, 23]

arr2 : [2, 3, 4, 5, 6]

Output array :

[ 1. 9. 0.5 4.2 3.83333333]

|  |  |
| --- | --- |
| **FUNCTION** | **DESCRIPTION** |
| [**add()**](https://www.geeksforgeeks.org/numpy-add-in-python/) | Add arguments element-wise. |
| **positive()** | Numerical positive, element-wise. |
| [**negative()**](https://www.geeksforgeeks.org/numpy-negative-in-python/) | Numerical negative, element-wise. |
| **multiply()** | Multiply arguments element-wise. |
| [**power()**](https://www.geeksforgeeks.org/numpy-power-python/) | First array elements raised to powers from second array, element-wise. |
| **subtract()** | Subtract arguments, element-wise. |
| [**true\_divide()**](https://www.geeksforgeeks.org/numpy-true_divide-python/) | Returns a true division of the inputs, element-wise. |
| [**floor\_divide()**](https://www.geeksforgeeks.org/numpy-floor_divide-python/) | Return the largest integer smaller or equal to the division of the inputs. |
| [**float\_power()**](https://www.geeksforgeeks.org/numpy-float_power-python/) | First array elements raised to powers from second array, element-wise. |
| **mod()** | Return the element-wise remainder of division. |
| **remainder()** | Return element-wise remainder of division. |
| **divmod()** | Return element-wise quotient and remainder simultaneously. |

**Complex number Function –**  
[**numpy.isreal(array)**](https://www.geeksforgeeks.org/numpy-isreal-python/)**:**Test element-wise whether it is a real number or not(not infinity or not Not a Number) and return the result as a boolean array.

|  |
| --- |
| # Python Program illustrating  # numpy.isreal() method    import numpy as geek    print("Is Real : ", geek.isreal([1+1j, 0j]), "\n")    print("Is Real : ", geek.isreal([1, 0]), "\n") |

Run on IDE

**Output :**

Is Real : [False True]

Is Real : [ True True]

[**numpy.conj(x[, out] = ufunc ‘conjugate’)**](https://www.geeksforgeeks.org/numpy-conj-in-python/)**:**This function helps the user to conjugate any complex number.  
The conjugate of a complex number is obtained by changing the sign of its imaginary part. If the complex number is 2+5j then its conjugate is 2-5j.

|  |
| --- |
| # Python3 code demonstrate conj() function    #importing numpy  import numpy as np    in\_complx1 = 2+4j  out\_complx1 = np.conj(in\_complx1)  print ("Output conjugated complex number of  2+4j : ", out\_complx1)    in\_complx2 =5-8j  out\_complx2 = np.conj(in\_complx2)  print ("Output conjugated complex number of 5-8j: ", out\_complx2) |

Run on IDE

**Output :**

Output conjugated complex number of 2+4j : (2-4j)

Output conjugated complex number of 5-8j: (5+8j)

**Special functions –**  
[**numpy.cbrt(arr, out = None, ufunc ‘cbrt’)**](https://www.geeksforgeeks.org/numpy-cbrt-python/)**:**This mathematical function helps user to calculate cube root of x for all x being the array elements.

|  |
| --- |
| # Python program explaining  # cbrt () function    import numpy as np    arr1 = [1, 27000, 64, -1000]  print ("cbrt Value of arr1 : \n", np.cbrt(arr1))    arr2 = [1024 ,-128]  print ("\ncbrt Value of arr2 : ", np.cbrt(arr2)) |

Run on IDE

**Output :**

cbrt Value of arr1 :

[ 1. 30. 4. -10.]

cbrt Value of arr2 : [ 10.0793684 -5.0396842]

[**numpy.clip()**](https://www.geeksforgeeks.org/numpy-clip-in-python/)**:**This function is used to Clip (limit) the values in an array.  
Given an interval, values outside the interval are clipped to the interval edges. For example, if an interval of [0, 1] is specified, values smaller than 0 become 0, and values larger than 1 become 1.

|  |
| --- |
| # Python3 code demonstrate clip() function    # importing the numpy  import numpy as np    in\_array = [1, 2, 3, 4, 5, 6, 7, 8 ]  print ("Input array : ", in\_array)    out\_array = np.clip(in\_array, a\_min = 2, a\_max = 6)  print ("Output array : ", out\_array) |

Run on IDE

**Output :**

Input array : [1, 2, 3, 4, 5, 6, 7, 8]

Output array : [2 2 3 4 5 6 6 6]

|  |  |
| --- | --- |
| **FUNCTION** | **DESCRIPTION** |
| **convolve()** | Returns the discrete, linear convolution of two one-dimensional sequences. |
| **sqrt()** | Return the non-negative square-root of an array, element-wise. |
| [**square()**](https://www.geeksforgeeks.org/numpy-square-python/) | Return the element-wise square of the input. |
| [**absolute()**](https://www.geeksforgeeks.org/numpy-absolute-python/) | Calculate the absolute value element-wise. |
| **fabs()** | Compute the absolute values element-wise. |
| **sign()** | Returns an element-wise indication of the sign of a number. |
| **interp()** | One-dimensional linear interpolation. |
| [**maximum()**](https://www.geeksforgeeks.org/numpy-maximum-in-python/) | Element-wise maximum of array elements. |
| [**minimum()**](https://www.geeksforgeeks.org/numpy-minimum-in-python/) | Element-wise minimum of array elements. |
| **real\_if\_close()** | If complex input returns a real array if complex parts are close to zero. |
| [**nan\_to\_num()**](https://www.geeksforgeeks.org/numpy-nan_to_num-in-python/) | Replace NaN with zero and infinity with large finite numbers. |
| **heaviside()** | Compute the Heaviside step function. |