Mathematical functions

In [2]:

```
import numpy as np
```

Trigonometric functions

Q1. Calculate sine, cosine, and tangent of x, element-wise.

In [3]:

```
x = np.array([0., 1., 30, 90])
print(np.sin(x))
print(np.cos(x))
print(np.tan(x))
```

```
[ 0.          0.84147098 -0.98803162  0.89399666]
[ 1.          0.54030231  0.15425145 -0.44807362]
[ 0.          1.55740772 -6.4053312  -1.99520041]
```

Q2. Calculate inverse sine, inverse cosine, and inverse tangent of x, element-wise.

In [4]:

```
x = np.array([-1., 0, 1.])
print(np.arcsin(x))
print(np.arccos(x))
print(np.arctan(x))
```

```
[-1.57079633  0.          1.57079633]
[3.14159265 1.57079633 0.        ]
[-0.78539816  0.          0.78539816]
```

Q3. Convert angles from radians to degrees.

In [5]:

```
x = np.array([-np.pi, -np.pi/2, np.pi/2, np.pi])
print(np.degrees(x))
```

```
[-180.  -90.   90.  180.]
```

Q4. Convert angles from degrees to radians.

In [6]:

```
x = np.array([-180.,  -90.,   90.,  180.])
print(np.radians(x))
```

```
[-3.14159265 -1.57079633  1.57079633  3.14159265]
```

In [8]:

```
x = np.array([-1., 0, 1.])
print(np.sinh(x))
print(np.cosh(x))
print(np.tanh(x))
```

```
[-1.17520119  0.          1.17520119]
[1.54308063 1.         1.54308063]
[-0.76159416  0.          0.76159416]
```

Rounding

Q6. Apply around(), floor(), ceil(), trunc()

In [9]:

```
x = np.array([2.1, 1.5, 2.5, 2.9, -2.1, -2.5, -2.9])
print(np.around(x))
print(np.floor(x))
print(np.ceil(x))
print(np.trunc(x))
```

```
[ 2.  2.  2.  3. -2. -2. -3.]
[ 2.  1.  2.  2. -3. -3. -3.]
[ 3.  2.  3.  3. -2. -2. -2.]
[ 2.  1.  2.  2. -2. -2. -2.]
```

Sums, products, differences

Q8. Apply sum(), prod(),cumsum(),min(),max(), mean() on both axises

In [12]:

```
x = np.array(
    [[1, 2, 3, 4],
     [5, 6, 7, 8]])
outs=[np.sum(x),
      np.sum(x,axis=0),
      np.sum(x,axis=1,keepdims=True),
      "",
      np.prod(x),
      np.prod(x,axis=0),
      np.prod(x,axis=1,keepdims=True),
      "",
      np.cumsum(x),
      np.cumsum(x,axis=0),
      np.cumsum(x,axis=1),
      "",
      np.min(x),
      np.min(x,axis=0),
      np.min(x,axis=1),
      "",
      np.max(x),
      np.max(x,axis=0),
      np.max(x,axis=1),
      "",
      np.mean(x),
      np.mean(x,axis=0),
      np.mean(x,axis=1)
      ]
for out in outs:
    if out == "":
        pass
        print
    else:
        pass
        print("->",out)
```

```
-> 36
-> [ 6  8 10 12]
-> [[10]
 [26]]
-> 40320
-> [ 5 12 21 32]
-> [[  24]
 [1680]]
-> [ 1  3  6 10 15 21 28 36]
-> [[ 1  2  3  4]
 [ 6  8 10 12]]
-> [[ 1  3  6 10]
 [ 5 11 18 26]]
-> 1
-> [1 2 3 4]
-> [1 5]
-> 8
-> [5 6 7 8]
-> [4 8]
-> 4.5
-> [3. 4. 5. 6.]
-> [2.5 6.5]
```

```
C:\Users\HP\Anaconda3\lib\site-packages\ipykernel_launcher.py:29: FutureWarn
ing: elementwise comparison failed; returning scalar instead, but in the fut
ure will perform elementwise comparison
```

Q9. Calculate the difference between neighboring elements, element-wise.

In [13]:

```
x = np.array([1, 2, 4, 7, 0])
print(np.diff(x))
```

```
[ 1  2  3 -7]
```

Q11. Return the cross product of x and y.

In [14]:

```
x = np.array([1, 2, 3])
y = np.array([4, 5, 6])

print(np.cross(x,y))
```

```
[-3  6 -3]
```

Exponents and logarithms

Q12. Compute e^x, element-wise.

In [15]:

```
x = np.array([1., 2., 3.], np.float32)

print(np.exp(x))
```

```
[ 2.7182817  7.389056  20.085537 ]
```

Q13. Calculate exp(x) - 1 for all elements in x.

In [16]:

```
x = np.array([1., 2., 3.], np.float32)
print(np.expm1(x))
```

```
[ 1.7182817  6.389056  19.085537 ]
```

Q14. Calculate 2^p for all p in x.

In [17]:

```
x = np.array([1., 2., 3.], np.float32)
print(np.exp2(x))
```

```
[2. 4. 8.]
```

Q15. Compute natural, base 10, and base 2 logarithms of x element-wise.

In [18]:

```
x = np.array([1, np.e, np.e**2])
print(np.log(x))
print(np.log10(x))
print(np.log2(x))
```

```
[0. 1. 2.]
[0.         0.43429448 0.86858896]
[0.         1.44269504 2.88539008]
```

Q16. Compute the natural logarithm of one plus each element in x in floating-point accuracy.

In [19]:

```
x = np.array([1e-99, 1e-100])
print(np.log1p(x))
```

```
[1.e-099 1.e-100]
```

Floating point routines

Q17. Return element-wise True where signbit is set.

In [20]:

```
x = np.array([-3, -2, -1, 0, 1, 2, 3])

print(np.signbit(x))
```

```
[ True  True  True False False False False]
```

Q18. Change the sign of x to that of y, element-wise.

In [140]:

```
x = np.array([-1, 0, 1])
y = -1.1
print(np.copysign(x))
```

```
[-1. -0. -1.]
```

Arithmetic operations

Q19. Add x and y element-wise.

In [22]:

```
x = np.array([1, 2, 3])
y = np.array([-1, -2, -3])
print(np.add(x,y))
```

```
[0 0 0]
```

Q20. Subtract y from x element-wise.

In [23]:

```
x = np.array([3, 4, 5])
y = np.array(3)

np.subtract(x,y)
```

Out[23]:

```
array([0, 1, 2])
```

Q21. Multiply x by y element-wise.

In [24]:

```
x = np.array([3, 4, 5])
y = np.array([1, 0, -1])

print(np.multiply(x,y))
```

```
[ 3  0 -5]
```

Q22. Divide x by y element-wise in two different ways.

In [25]:

```
x = np.array([3., 4., 5.])
y = np.array([1., 2., 3.])
print(np.true_divide(x,y))
print(np.floor_divide(x,y))
```

```
[3.         2.         1.66666667]
[3. 2. 1.]
```

Q23. Compute numerical negative value of x, element-wise.

In [26]:

```
x = np.array([1, -1])
print(np.negative(x))
```

```
[-1  1]
```

Q24. Compute the reciprocal of x, element-wise.

In [27]:

```
x = np.array([1., 2., .2])
print(np.reciprocal(x))
```

```
[1.  0.5 5. ]
```

Q25. Compute x^y, element-wise.

In [28]:

```
x = np.array([[1, 2], [3, 4]])
y = np.array([[1, 2], [1, 2]])
print(np.power(x,y))
```

```
[[ 1  4]
 [ 3 16]]
```

Q26. Compute the remainder of x / y element-wise in two different ways.

In [29]:

```
x = np.array([-3, -2, -1, 1, 2, 3])
y = 2
print(np.mod(x,y))
print(np.fmod(x,y))
```

```
[1 0 1 1 0 1]
[-1  0 -1  1  0  1]
```

Miscellaneous

Q27. If an element of x is smaller than 3, replace it with 3. And if an element of x is bigger than 7, replace it with 7.

In [30]:

```
x = np.arange(10)
print(np.clip(x,3,7))
```

```
[3 3 3 3 4 5 6 7 7 7]
```

Q28. Compute the square of x, element-wise.

In [32]:

```
x = np.array([1, 2, -1])
print(np.square(x))
```

```
[1 4 1]
```

Q29. Compute square root of x element-wise.

In [33]:

```
x = np.array([1., 4., 9.])
print(np.sqrt(x))
```

```
[1. 2. 3.]
```

Q30. Compute the absolute value of x.

In [34]:

```
x = np.array([[1, -1], [3, -3]])

print(np.abs(x))
```

```
[[1 1]
 [3 3]]
```

Q31. Compute an element-wise indication of the sign of x, element-wise.

In [35]:

```
x = np.array([1, 3, 0, -1, -3])
print(np.sign(x))
```

```
[ 1  1  0 -1 -1]
```

In [ ]: