In [17]:
```
len(range(1000,1000000))
```

Out[17]:
```
999000
```

In [24]:
```
import numpy as np

height=[1,10,20,30,40,50]
delta=[1,1,2,3,4,5]

height1=range(1000,1000000)
delta1=range(1,999001)
```

In [19]:
```
def update_height(height,delta):

    height = np.array(height)

    delta = np.array(delta)

    new_height = height+delta

    return new_height
```

In [20]:
```
def update_height_python(height, delta):

    new_height=[]

    for i in range(len(height)):

        new_height.append(height[i]+delta[i])

    return new_height
```

In [21]:
```
%timeit update_height(height,delta)
```

```
2.07 µs ± 26.9 ns per loop (mean ± std. dev. of 7 runs, 100000 loops each)
```

In [22]:
```
%timeit update_height_python(height,delta)
```

```
941 ns ± 6.48 ns per loop (mean ± std. dev. of 7 runs, 1000000 loops each)
```

In [25]:
```
%timeit update_height(height1,delta1)
```

```
136 ms ± 1.12 ms per loop (mean ± std. dev. of 7 runs, 10 loops each)
```

In [26]:
```
%timeit update_height_python(height1,delta1)
```

```
242 ms ± 4.88 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
```

In [27]:
```
ord("a")
```

Out[27]:
```
97
```

In [28]:
```
ord("A")
```

Out[28]:
```
65
```

In [29]:
```
a=np.arange(1,10).reshape((3,3))
a
```

Out[29]:
```
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])
```

In [31]:
```
np.rot90(a,k=-1)
```

Out[31]:
```
array([[7, 4, 1],
       [8, 5, 2],
       [9, 6, 3]])
```

In [32]:
```
np.rot90(a,k=-2)
```

Out[32]:
```
array([[9, 8, 7],
       [6, 5, 4],
       [3, 2, 1]])
```

In [33]:
```
np.rot90(a,k=1)
```

Out[33]:
```
array([[3, 6, 9],
       [2, 5, 8],
       [1, 4, 7]])
```

In [ ]:
```
tf.
```

In [34]:
```
X = np.arange(12).reshape(3,4)
```

In [35]:
```
X
```

Out[35]:
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

In [37]:
```
X.sum()=>11
```

```
  File "/var/folders/hd/9z4dczb56dj54lb7q8w7s4zw0000gn/T/ipykernel_1794/3032116454.py", line 1
    X.sum()=>11
           ^
SyntaxError: cannot assign to function call
```

In [ ]:
```
x = np.ones((5,5))
x[1:-1,1:-1] = 0
```

In [38]:
```
np.ones((5,5))
```

Out[38]:
```
array([[1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1.]])
```

In [39]:
```
np.zeros((5,5))
```

Out[39]:
```
array([[0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0.]])
```

In [ ]:
```

```

In [42]:
```
arr=np.arange(1,10).reshape((3,3))
arr
```

Out[42]:
```
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])
```

In [46]:
```
arr[:, 0]
```

Out[46]:
```
array([1, 4, 7])
```

In [43]:
```
np.hstack((arr,arr[:, 0])) --> Throwing an error
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
/var/folders/hd/9z4dczb56dj54lb7q8w7s4zw0000gn/T/ipykernel_1794/3266422096.py in <module>
---> 1 np.hstack((arr,arr[:, 0]))

<__array_function__ internals> in hstack(*args, **kwargs)

~/opt/anaconda3/lib/python3.9/site-packages/numpy/core/shape_base.py in hstack(tup)
    344         return _nx.concatenate(arrs, 0)
    345     else:
--> 346         return _nx.concatenate(arrs, 1)
    347
    348

<__array_function__ internals> in concatenate(*args, **kwargs)

ValueError: all the input arrays must have same number of dimensions, but the array at index 0 has 2 dimension(s) and the array at index 1 has 1 dimension(s)
```

In [47]:
```
arr[:, [0]]
```

Out[47]:
```
array([[1],
       [4],
       [7]])
```

In [ ]:
```
np.hstack((arr, arr[:, [0]])).shape= (4, 3)
```

In [ ]:
```
np.hstack((arr, arr[:, 0])).shape = (3, 4
```

In [48]:
```
np.hstack((arr, arr[:, [0]]))
```

Out[48]:
```
array([[1, 2, 3, 1],
       [4, 5, 6, 4],
       [7, 8, 9, 7]])
```

In [ ]:
```
a[:, 0]
```

In [49]:
```
a=np.arange(1,9).reshape((4,2))
a
```

Out[49]:
```
array([[1, 2],
       [3, 4],
       [5, 6],
       [7, 8]])
```

In [50]:
```
b=np.zeros((6,4))
b
```

Out[50]:
```
array([[0., 0., 0., 0.],
       [0., 0., 0., 0.],
       [0., 0., 0., 0.],
       [0., 0., 0., 0.],
       [0., 0., 0., 0.],
       [0., 0., 0., 0.]])
```

In [52]:
```
b[1:-1,1:-1]=a
```

In [53]:
```
b
```

Out[53]:
```
array([[0., 0., 0., 0.],
       [0., 1., 2., 0.],
       [0., 3., 4., 0.],
       [0., 5., 6., 0.],
       [0., 7., 8., 0.],
       [0., 0., 0., 0.]])
```

In [54]:
```
np.random.randint(1,10,10)
```

Out[54]:
```
array([7, 2, 7, 6, 9, 1, 4, 9, 9, 9])
```

In [55]:
```
np.random.rand()
```

Out[55]:
```
0.9564415233508502
```

In [60]:
```
np.random.normal(5,1,100)
```

Out[60]:
```
array([7.19723226, 5.36244027, 3.44420316, 6.08054765, 4.74148946,
       4.68000732, 4.65283999, 4.36851552, 5.54706577, 5.45424251,
       4.25513455, 4.26866565, 5.80926296, 3.96511915, 4.31234961,
       4.2199754 , 4.5763019 , 4.95576587, 3.11979707, 5.5643522 ,
       3.9558589 , 6.2486203 , 6.65036409, 5.46762961, 5.56413901,
       5.82137626, 5.0844992 , 6.354304  , 3.47934381, 3.87997716,
       6.21083741, 8.75756013, 5.69905266, 4.51661355, 5.51482516,
       5.35567133, 5.61325824, 5.45830344, 4.58439867, 4.07949802,
       4.3447844 , 3.77391503, 4.76327062, 5.89075655, 6.26975299,
       5.73771262, 4.77302395, 4.7144148 , 3.05743142, 4.60837514,
       5.74984715, 4.37764688, 4.41383785, 4.85640053, 6.98039468,
       4.56235284, 4.68876413, 5.18947069, 4.13276002, 5.50461059,
       5.44274399, 5.17758122, 2.23992147, 4.80873395, 5.9932737 ,
       3.09104118, 5.09201833, 4.63836811, 3.34779365, 4.40724708,
       4.27772221, 5.84415988, 4.79929948, 5.06261629, 5.15418651,
       4.52266157, 6.02241394, 6.00980542, 4.23583077, 5.1011585 ,
       6.55842677, 7.1991893 , 4.28764179, 5.50172341, 4.59010286,
       5.65874295, 5.03862594, 5.48089835, 5.15382741, 4.87910736,
       5.39653359, 6.10529329, 4.69049386, 7.03417551, 6.0855221 ,
       3.02908563, 3.97594732, 4.10052173, 5.3753579 , 5.61417264])
```

In [59]:
```

```

In [ ]:
```

```