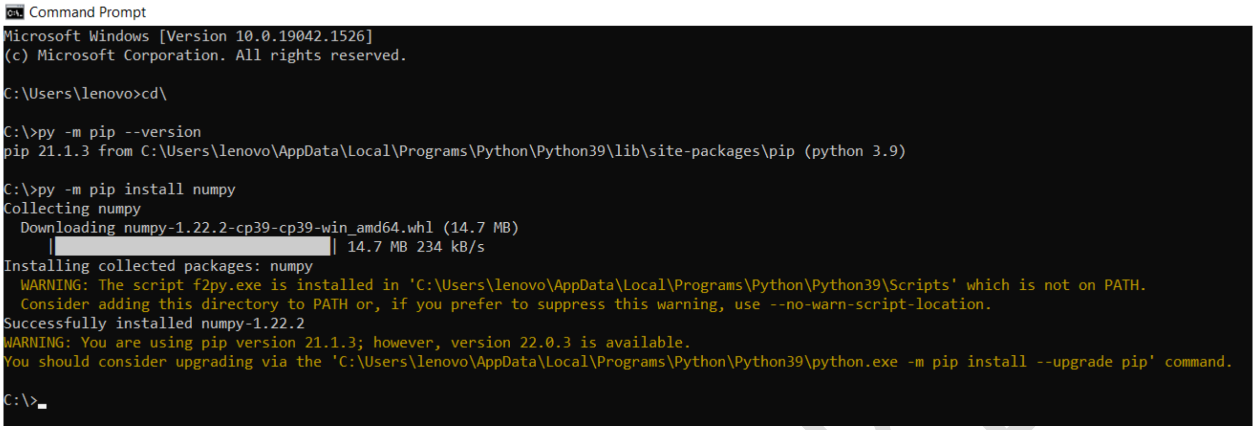
1. Install Numpy

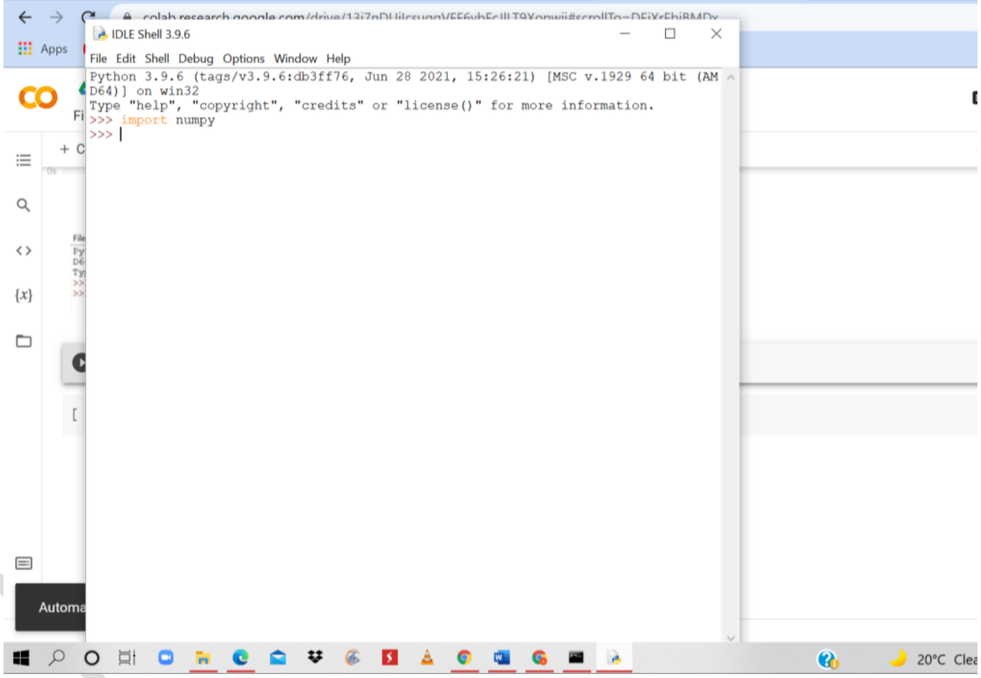
Ans:



```
Microsoft Windows [Version 10.0.19042.1526]
(c) Microsoft Corporation. All rights reserved.

C:\Users\lenovo>cd\

C:\>py -m pip --version
pip 21.1.3 from C:\Users\lenovo\AppData\Local\Programs\Python\Python39\lib\site-packages\pip (python 3.9)

C:\>py -m pip install numpy
Collecting numpy
  Downloading numpy-1.22.2-cp39-cp39-win_amd64.whl (14.7 MB)
     |████████████████████████████████| 14.7 MB 234 kB/s
Installing collected packages: numpy
  WARNING: The script f2py.exe is installed in 'C:\Users\lenovo\AppData\Local\Programs\Python\Python39\Scripts' which is not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
Successfully installed numpy-1.22.2
WARNING: You are using pip version 21.1.3; however, version 22.0.3 is available.
You should consider upgrading via the 'C:\Users\lenovo\AppData\Local\Programs\Python\Python39\python.exe -m pip install --upgrade pip' command.

C:\>_
```

2. Check the Numpy version installed

Ans:



```
Python 3.9.6 (tags/v3.9.6:db3ff76, Jun 28 2021, 15:26:21) [MSC v.1929 64 bit (AM
D64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> import numpy
>>> |
```

3. Create 1-D Array in numpy:

Ans:

```
import numpy as np
a1=np.array([1,2,3,4])
print(a1)
```

Output:

```
[1 2 3 4]
```

4. Use list to create 1D array (you may also specify data type i.e. dtype=’int16’)

Ans:

```
import numpy as np
a2=np.array([6,7,9,10],dtype='int16')
print(a2)
```

Output:

```
[ 6  7  9 10]
```

5. User tuple to create 1D array

Ans:

```
import numpy as np
a3=np.array(('a','b','c','d'))
print(a3)
```

Output:

```
['a' 'b' 'c' 'd']
```

6. Use arange function to create 1D array of int

Ans:

```
import numpy as np
a4=np.arange(1,10,2)

print(a4)
```

Output:

```
[1 3 5 7 9]
```

7. Use arange function to create 1D array of float (may use dtype = symbols(int->‘i’, uint >’u’,float->’f’,double->’d’,complex->’D’,bool->’b’)

Ans:

```
import numpy as np
a5=np.arange(1.0,10.0,2)
print(a5)
```

Output:

```
[1. 3. 5. 7. 9.]
```

8. Create 1D array of mixed elements int and float, and print the array and see the output

Ans:

```
import numpy as np
a6=np.array((1,2,3.4,8,1.5))
print(a6)
```

Output:

```
[1.  2.  3.4 8.  1.5]
```

9. Create 1D array of mixed elements int, float, and str, then print the array and see the output

Ans:

```
import numpy as np
a7=np.array((1,2,3.4,'a',1.5,'i'))
print(a7)
```

Output:

```
['1' '2' '3.4' 'a' '1.5' 'i']
```

10. Create a 2D array of dimensions 2x2

Ans:

```
m1=np.zeros((2,2),dtype=int)
print(m1)
```

Output:

```
[[0 0]
 [0 0]]
```

11. Print the shape, size, and memory used by this array in bytes (use itemsize, or nbytes)

Ans:

```
print(m1.shape)
print(m1.size)
print(m1.size * m1.itemsize)
```

Output:

```
(2, 2)
4
32
```

12. Check the type of any array variable

Ans: `print(type(m1))`

Output:

```
<class 'numpy.ndarray'>
```

13. Check indexing on array with help of examples

Ans:

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

Output:

```
4
```

14. Using arange function create an 3D array of dimensions = (2,3,4) , first element of this array is 0 and last element is 23 in increasing order, store this array in a variable b.

Ans:

```
b=np.arange(0,24,1)

b=b.reshape(2,3,4)
print(b)
```

Output:

```
[[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]]

[[12 13 14 15]
 [16 17 18 19]
[20 21 22 23]]]
```

15. What index can produce output: array([[0, 1, 2, 3], [4, 5, 6, 7], [8, 9, 10, 11]])

Ans:

```
print(b[0])
```

Output:

```
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]]
```

16. What index can produce output: 0

Ans:

```
print(b[0][0][0])
```

Output:

```
0
```

17. What index can produce output: array([4, 5, 6, 7])

Ans:

```
print(b[0][1])
```

Output:

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

18. What index can produce output: array([0,12])

Ans: print(b[:,0:4:3,0])

Output:

```
[[ 0]
[12]]
```

19. What index can produce output: array([4,6])

Ans: print(b[0,1,0:3:2])

Output:

```
[4 6]
```

20. Check the output of b[… , 1]

Ans: print(b[...,1])

Output:

```
[[ 1  5  9]
[13 17 21]]
```

21. What index can produce output: array([1, 5, 9])

Ans: print(b[0,:,1])

Output: [1 5 9]

22. What index can produce output: array([3,7,11])

Ans: print(b[0,:,3])

Output:[3 7 11]

23. What index can produce output: array([11, 7,3])

Ans: print(b[0,::-1,3])

Output: [11 7 3]

24. What index can produce output: array([3,11])

Ans: print(b[0,0:3:2,3])

Output: [3 11]

25. Use function ravel() with array b, and observe the output

Ans: print(b.ravel())

Output:
```
[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23]
```

26. Use function flatten() with array b, and observe the output

Ans: print(b.flatten())

Output:
```
[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23]
```

27. Use function transpose() with array b, check the output

Ans: print(b.transpose())

Output:
```
[[[ 0 12]
  [ 4 16]
  [ 8 20]]

 [[ 1 13]
 [ 5 17]
 [ 9 21]]

 [[ 2 14]
 [ 6 18]
 [10 22]]

 [[ 3 15]
 [ 7 19]
 [11 23]]]
```

28. Use function T() with array b, check the output

Ans: print(b.T)

Output:

```
[[[ 0 12]
[ 4 16]
[ 8 20]]

[[ 1 13]
[ 5 17]
[ 9 21]]

[[ 2 14]
[ 6 18]
[10 22]]

[[ 3 15]
[ 7 19]
```

```
 [11 23]]]
```

29. Use function concatenate on two arrays with axis 0, and axis 1 and observe the output: i.e np.concatenate((arr1,arr2),axis=0) & np.concatenate((arr1,arr2),axis=1)

Ans:

```
arr1=np.array([[1,2,3],[4,5,6],[7,8,9]])

arr2=np.array([[10,11,12],[13,14,15],[16,17,18]])
c=np.concatenate((arr1,arr2),axis=0)
print(c)
```

Output:

```
[[ 1  2  3]
 [ 4  5  6]
 [ 7  8  9]
 [10 11 12]
 [13 14 15]
 [16 17 18]]
```

```
arr1=np.array([[1,2,3],[4,5,6],[7,8,9]])
 arr2=np.array([[10,11,12],[13,14,15],[16,17,18]])
d=np.concatenate((arr1,arr2),axis=1)
print(d)
```

Output:

```
[[ 1  2  3 10 11 12]
 [ 4  5  6 13 14 15]
 [ 7  8  9 16 17 18]]
```

30. Use function astype() to convert the array to an array on another type A=array([[0, 1],[2, 3],[4, 5]]) A.astype(float) or A.astype(‘f’) or A.astype(‘float64’) A.astype(bool)

Ans:

```
A=np.array([[0, 1], [2, 3],[4, 5]])

print(A.astype(float))
print(A.astype(bool))
```

Output:

```
[[0. 1.]
 [2. 3.]
 [4. 5.]]

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

31. Check the output of “ np.eye(3)” and “np.zeros(3)”

Ans:

```
B=np.eye(3)

print(B)
Output:
[[1. 0. 0.]
[0. 1. 0.]
[0. 0. 1.]]

C=np.zeros(3)
print(C)
```

Output:

```
[0. 0. 0.]
```

32. Find minimum, maximum, and average of an array

Ans:

```
A=np.array([0,1,2,3,4,5,6,7])
print(A.min())
print(A.max())
print(np.average(A))
```

Output:

```
0
7
3.5
```

33. Find matrix multiplication user dot function

Ans:

```
arr1=np.array([[0,1,2],[3,4,5],[6,7,8]])

arr2=np.array([[6,7,8],[3,4,5],[0,1,2]])
ans=np.dot(arr1,arr2)
print(ans)
```

Output:

```
[[ 3  6  9]
 [30 42 54]
[57 78 99]]
```

34. Find element wise multiplication of two matrices using multiply function

Ans:

```
arr1=np.array([[0,1,2],[3,4,5],[6,7,8]])
```

```
arr2=np.array([[6,7,8],[3,4,5],[0,1,2]])
ans=np.multiply(arr1,arr2)
print(ans)
```

Output:

```
[[ 0  7 16]
[ 9 16 25]
[ 0  7 16]]
```

35. Check sctypeDict.keys() and note down the info which you understand properly.

Ans: A (truncated) list of all the full data type codes can be found by

applying sctypeDict.keys()

36. Creating new dtype: record=np.dtype([('name',str,40),('no_of_items',int),('price','float64')])

Ans:

```
record=np.dtype([('name',str,40),('no_of_items',int),('price','float64')])
print(np.array([1,3,2,4],dtype=record))
```

Output:

```
[('1', 1, 1.) ('3', 3, 3.) ('2', 2, 2.) ('4', 4, 4.)]
```

37. Store record type values in variable items,

Ans:

```
item=np.array([1,3,2,4],dtype=record)
item
```

Output:

```
array([('1', 1, 1.), ('3', 3, 3.), ('2', 2, 2.), ('4', 4, 4.)],
  dtype=[('name', '<U40'), ('no_of_items', '<i8'), ('price', '<f8')])
```

38. items=np.array([('life of dvd',42,30.50),('Butter',10,22.75)], dtype=record)

Ans:

```
items=np.array([('life of dvd',42,30.50),('Butter',10,22.75)], dtype=record)
I          tems
```

Output:

```
array([('life of dvd', 42, 30.5 ), ('Butter', 10, 22.75)],
      dtype=[('name', '<U40'), ('no_of_items', '<i8'), ('price', '<f8')])
```

39.Multiplication using @

Ans:

```
arr1=np.array([[0,1,2],[3,4,5],[6,7,8]])

arr2=np.array([[6,7,8],[3,4,5],[0,1,2]])
ans=arr1@arr2
print(ans)
```

Output:

```
[[ 3  6  9]

 [30 42 54]
 [57 78 99]]
```

40.Multiplication using cross.

Ans:

```
arr1=np.array([[0,1,2],[3,4,5],[6,7,8]])
ans=np.cross(arr1,arr1)
print(ans)
```

Output:

```
[[0 0 0]
 [0 0 0]
 [0 0 0]]
```

41.Multiplication using *.

Ans:

```
arr1=np.array([[0,1,2],[3,4,5],[6,7,8]])

arr2=np.array([[6,7,8],[3,4,5],[0,1,2]])
ans=arr1*arr2
print(ans)
```

Output:

```
[[ 0  7 16]

 [ 9 16 25]
 [ 0  7 16]]
```